Trabajo Fin de Grado
Ingeniería Aeroespacial

Optimización de trayectorias orbitales interplanetarias mediante algoritmos metaheurísticos

Autor: Azael Superviel Ferrer

Tutor: Francisco Gordillo Álvarez

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Ingeniería de Sistemas y Automática

Trabajo Fin de Grado
Ingeniería Aeroespacial

Optimización de trayectorias orbitales interplanetarias mediante algoritmos metaheurísticos

Autor:
Azael Superviel Ferrer

Tutor:
Francisco Gordillo Álvarez
Profesor Titular

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado: Optimización de trayectorias orbitales interplanetarias mediante algoritmos metaheurísticos

Autor: Azael Superviel Ferrer
Tutor: Francisco Gordillo Álvarez

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Resumen

En el campo de la mecánica orbital, donde está incluido el diseño de misiones interplanetarias, los costes de ejecución de una misión son disparatados, luego la optimización es un pilar fundamental a la hora de reducirlos. Es por ello que en este trabajo se pretende optimizar varios problemas referentes a dos misiones reales (Cassini o Messenger) y un problema ficticio ideado por la ESA (GTOC1) utilizando algoritmos de optimización metaheurísticos como son los algoritmos genéticos y los algoritmos de optimización por enjambre de partículas. El uso de estos algoritmos se debe a la complejidad de los problemas que se tratan y la dificultad que supone optimizarlos mediante otros métodos tradicionales.

Puesto que el modelado de una misión completa y su optimización correspondiente no se realiza hasta las fases finales del diseño de una misión, en este trabajo se han utilizado modelos que incluyen ciertas simplificaciones para poder llevar a cabo las optimizaciones con ordenadores alcanzables para cualquier persona sin la necesidad de utilizar supercomputadores.

Abstract

In the field of orbital mechanics, where the design of interplanetary missions is included, the execution costs of a mission are huge, therefore optimization is a fundamental pillar when it comes to reducing them. That is why this work aims to optimize several problems related to two real missions (Cassini or Messenger) and a fictitious problem devised by ESA (GTOC1) using metaheuristic optimization algorithms such as genetic algorithms and particle swarm optimization algorithms. The use of these algorithms is due to the complexity of the problems and the difficulty to optimize them using other traditional optimization methods.

Due to the fact that the modeling of a complete mission and its corresponding optimization is not carried out until the final stages of the design of a mission, in this work models that include simplifications have been used to carry out the optimizations with computers that are achievable for anyone without the need for supercomputers.

Índice

Notación

GA	Algoritmos genéticos
PSO	Algoritmos de optimización por enjambre de partículas
J	Función objetivo
x	Vector de decisión
g	Restricciones del problema
χ	Espacio de búsqueda
p_i	Variables que componen el vector de decisión
N	Número de variables que componen el vector de decisión
$pos_i(t)$	Posición de la partícula i en la iteración t
$v_i(t)$	Velocidad de la partícula i en la iteración t
ω	Factor de inercia en PSO o argumento de periapsis en elementos orbitales
r_1 y r_2	Parámetros aleatorios en PSO
c_1	Constante de atracción al mejor personal en PSO
c_2	Constante de atracción al mejor global en PSO
p_i^{mejor}	Mejor valor encontrado por la partícula i
p_g^{mejor}	Mejor valor encontrado por el enjambre en su conjunto
$\mathbf{r}$	Vector posición de un objeto
$\dot{\mathbf{x}}$	Derivada respecto del tiempo del vector x
ε	Energía específica
$\mathbf{h}$	Vector momento cinético
$\mathbf{e}$	Vector excentricidad
a	Semieje mayor de una órbita
e	Excentricidad de una órbita
Ω	Ascensión recta del nodo ascendente
i	Inclinación de la órbita
θ	Anomalía verdadera
Υ	Primer punto de Aries
V_i	Velocidad inicial en una maniobra orbital
V_f	Velocidad final en una maniobra orbital
ΔV	Impulso en una maniobra orbital
$\odot$	El Sol
$\oplus$	La Tierra
☿	Mercurio
♀	Venus
♃	Júpiter

♄	Saturno
t_0	Fecha de comienzo de la misión
V_∞, u y v	Definen la velocidad heliocéntrica a la salida de la hipérbola planetocéntrica
T_N	Tiempo de vuelo entre los planetas N y $N-1$ de la secuencia de planetas sobrevolados
η_N	Fracción del viaje en la que se realiza la maniobra en espacio profundo
r_{P_N}	Radio mínimo de acercamiento al planeta N
b_{incl_N}	Ángulo de rotación de la velocidad de escape de una órbita hiperbólica planetocéntrica respecto a la velocidad de entrada en ella.

1 Introducción

Dos ramas científico-técnicas que están en pleno auge y que tomarán mucha importancia en los años próximos son la investigación y desarrollo de inteligencias artificiales y la exploración del espacio que nos rodea mediante la realización de proyectos de misiones interplanetarias o de misiones cuyo objetivo es salir del Sistema Solar. En este trabajo se pretende complementar dichas ramas.

Como se sabe, la inteligencia artificial es un área de la ciencia de la computación enfocada a la creación de máquinas que trabajen y actúen como humanos. Algunas de las funciones que se han conseguido por el momento son: la capacidad de mantener un diálogo, de aprender por sí mismas o de resolver problemas entre otras. En esta última función es en la que se va a profundizar y trabajar en este proyecto. Una de las razones por las que se han utilizado algunos algoritmos de inteligencia artificial para la resolución de problemas complejos es que se ha convertido en una parte esencial de la propia industria y puede suponer una cierta ventaja el conocer de qué trata este área de conocimiento.

Del mismo modo, la exploración espacial tripulada es un campo que vuelve a estar a la orden del día ya que desde la última misión a la Luna, *Apollo* 17, los humanos únicamente han salido de la atmósfera baja terrestre para alcanzar la Estación Espacial Internacional. Actualmente, ya están en desarrollo algunas misiones tripuladas a la Luna y a Marte. Además, se ha disparado el número de misiones no tripuladas desarrolladas por empresas privadas en conjunto con las grandes agencias espaciales del mundo (ESA, NASA, JAXA, etc.). Uno de los procesos importantes en los proyectos de misiones espaciales es la optimización de las mismas en función de los parámetros de mayor interés: tiempo, coste, etc. Por ejemplo, una optimización en combustible puede provocar un ahorro considerable en costes de lanzamiento y fabricación de la nave espacial o puede permitir sustituir ese peso ahorrado por peso de instrumentación científica que dé lugar a una mejor investigación durante la misión.

1.1 Alcance y estructura del trabajo

Debido al desarrollo actual y la importancia que van a tener en un futuro próximo los dos campos comentados se ha decidido enfocar el trabajo de manera que se combinen ambos. El objetivo principal del trabajo es comprobar la viabilidad de dos algoritmos metaheurísticos para resolver problemas de optimización global de misiones interplanetarias reales con ciertas simplificaciones que permiten abordar la optimización sin la necesidad de un supercomputador y facilitando la optimización frente a los métodos más tradicionales. De manera general, la optimización global de un problema trata de encontrar un punto x en un espacio de búsqueda limitado por una serie de restricciones y en el que una función objetivo contiene un máximo o un mínimo. Uno de los aspectos complicados de este tipo de optimización es que las funciones que se estudian suelen ser muy

complejas y poseen una gran cantidad de máximos o mínimo locales que evitan que los algoritmos de optimización encuentren fácilmente el máximo o mínimo global que se busca. Aunque es cierto que los algoritmos utilizados tienen métodos para "escapar" de estos máximos o mínimos locales, a veces no son suficientes y los resultados que se generan no corresponden al que realmente se busca.

En cuanto a la estructura por la que se ha optado, puede comentarse que se ha dividido el trabajo por capítulos en los cuales se expone todo lo que se debe conocer hasta llegar a los resultados de las optimizaciones realizadas.

En el Capítulo 2, se tratan de explicar los dos algoritmos basados en inteligencia artificial que se han aplicado para resolver los problemas de optimización: el algoritmo genético y el de optimización por enjambre de partículas. Añadiendo conceptos básicos y estructura general de cada uno de ellos.

En el Capítulo 3, se introducen los conceptos de la mecánica orbital que se implementan en los códigos de modelización de los diferentes problemas.

En el Capítulo 4, se exponen los problemas a resolver mostrando la función objetivo, las variables de las que depende la optimización y las restricciones que se imponen para el vector de decisión según se tengan maniobras en espacio profundo o no.

En el Capítulo 5, se explica el proceso de obtención de los valores para cada uno de los parámetros que influyen en el funcionamiento de los algoritmos de optimización utilizados junto con los resultados obtenidos y la elección final.

En el Capítulo 6, se añaden los resultados a los que se ha llegado tras las optimizaciones de los problemas a tratar con los métodos de optimización comentados. Introduciendo gráficas de la evolución de los mejores valores encontrados para cada generación o iteración del algoritmo y tablas comparativas de mejores resultados y tiempos para comparar el funcionamiento de los diferentes algoritmos en cada problema.

En el Capítulo 7, se cierra el trabajo comentando las conclusiones a las que se ha llegado durante la realización del mismo y las posibles mejoras futuras que podrían aplicarse en el caso de ampliar y complementar los estudios que aquí se han ejecutado.

Tras la memoria principal, se añaden dos apéndices en los que se exponen: en primer lugar, el comportamiento del algoritmo genético en las generaciones finales de la optimización y, en segundo lugar, los algoritmos utilizados para las optimizaciones.

2 Algoritmos de inteligencia artificial aplicados a la optimización utilizados en este trabajo

Desde la mitad del siglo XX ha habido científicos que han centrado sus estudios en la inteligencia artificial y en tratar de conseguir programas computacionales que simularan un cierto comportamiento natural para la resolución de problemas de todo tipo. En este capítulo se presentan por secciones dos de los algoritmos más importantes en lo que respecta a la optimización global heurística, comentando su origen, los conceptos requeridos para su entendimiento y su estructura.

2.1 Algoritmos genéticos

El algoritmo genético es un tipo de algoritmo de optimización que fue mencionado por primera vez por John Holland en su libro *Adaptation in Natural and Aritificial Systems* de 1975. Lo que se expuso en dicho libro se convirtió en la base de un concepto mucho más amplio que se fue desarrollando con el paso de los años hasta nuestros días [11].

Pese a que John Holland acuñó el término de «algoritmo genético», durante los años 60 hubo científicos de Alemania (Ingo Rechenberg y Hans-Paul Schwefel) y de Estados Unidos (Bremermann y Fogel) que consiguieron introducir algunas ideas, tomando como referencia la teoría de la evolución darwinista, que estaban relacionadas con la mutación y selección de los genes durante la creación de las generaciones posteriores. Tras la publicación de John Holland se multiplicó la investigación en el campo de la metaheurística y fueron algunos de sus alumnos (Ken DeJong y David Goldberg) los que siguieron perfeccionando este concepto.

La aplicación de este algoritmo se basa en encontrar un máximo o un mínimo de una cierta función objetivo relacionada con un problema computacional. Para ello genera una población inicial aleatoriamente (que consiste en un conjunto de posibles soluciones del problema) y, basándose en la teoría de la evolución, tiene la capacidad de originar nuevas generaciones. Cada individuo de la población total tendrá asociado un valor de la función objetivo, por lo que la finalidad es obtener un cierto número de generaciones imitando los procesos biológicos de reproducción y selección natural para dar como resultado el individuo que tenga asociado el mejor valor de la función objetivo. Puesto que la mayoría de procesos que se llevan a cabo durante la ejecución del agoritmo son aleatorios, es posible su aplicación a cualquier tipo de función sin importar su continuidad, derivabilidad, etc. Dotando de una gran eficiencia y aplicabilidad a este método frente a los clásicos de optimización y

provocando así su amplio uso en el campo de la heurística.

En lo que sigue se van a comentar algunos conceptos básicos del algoritmo junto con su estructura general y posteriormente se explicarán en detalle los conceptos añadiendo cómo las funciones de MATLAB, que se han utilizado para la obtención de resultados, incluyen estos conceptos en sus líneas.

2.1.1 Conceptos básicos y estructura

Debido a que el algoritmo se basa en diferentes procesos de la evolución biológica gran parte de la terminología se toma prestada de la biología. Los principales conceptos a tener en cuenta para entender y poder realizar una aplicación del algoritmo génetico son:

- La función objetivo y vector de decisión
- La población de cromosomas
- La selección de cromosomas que van a reproducirse
- El cruce de los cromosomas escogidos para dar lugar a una nueva generación
- La mutación aleatoria en las nuevas generaciones
- El elitismo

La función objetivo es la función que se pretende optimizar ($J(x)$). Para ello se parte de un cierto espacio de búsqueda (χ) donde la función está definida y el algoritmo trata de encontrar la solución óptima para dicho problema, que por lo general será:

$$\min_{x \in \chi} \; J(x), \tag{2.1}$$

donde x es el vector de decisión, el cual representa a un conjunto de valores asociados a unas ciertas variables necesarias para la evaluación de la función objetivo y que lleva asociado un único valor de esta función.

Respecto a los cromosomas que conforman la población no son más que diferentes vectores de decisión que han sido generados aleatoriamente durante la ejecución del algoritmo y que, como se ha comentado, proporcionan un valor de la función objetivo. Dicho valor proporcionado permite una mejor selección de los cromosomas que van a reproducirse, ya que es posible asignarle una mayor probabilidad de selección a aquellos con mejores valores de la función objetivo, para que, durante el proceso de reproducción, estos cromosomas sean los que principalmente tengan descendencia y así las nuevas generaciones puedan tener asociados valores más cercanos al óptimo de la función. Para un problema de dimensión N, un cromosoma puede expresarse como:

$$x = [p_1, p_2, ..., p_N] \tag{2.2}$$

siendo p_i cada una de las variables de las que dependerá la optimización.

En cuanto al proceso de generación de nuevos cromosomas, habrá que tener presente el método de cruce de los cromosomas progenitores. Este método suele partir los cromosomas progenitores por un cierto lugar e intercambiar una de las partes entre ambos dando lugar a dos descendientes con cromosomas fusionados de los anteriores. A dicha fusión se le puede incluir la posible aparición de una mutación como si de la propia evolución biológica se tratase. Dicha mutación suele tener una probabilidad baja de aparición y permite que alguna de las componentes (genes) que conforman

los cromosomas cambie aleatoriamente. A continuación se expone un ejemplo de reproducción de cromosomas, comenzando por los padres:

$$[aabccb|cacb] \quad \text{y} \quad [bbccab|acab] \tag{2.3}$$

En este caso se ha decidido que los cromosomas se dividan en seis y cuatro componentes cada uno, por lo que la descendencia de estos podría ser:

$$[aaaccb|acab] \quad \text{y} \quad [bbccab|bacb] \tag{2.4}$$

Como puede verse se intercambian las cadenas de caracteres en las que se dividen los padres pero además es posible que aparezca la mutación en alguna de las componentes. La mutación ayuda a que el algoritmo no converja rápidamente hacia un mínimo local y obtenga otros resultados del espacio de búsqueda que se acerquen al óptimo global.

Este tipo de algoritmo suele trabajar hasta alcanzar un número máximo de generaciones que se le haya impuesto o hasta que el mejor punto encontrado a lo largo de varias iteraciones se haya estabilizado. Además, es posible conseguir que el programa genere soluciones más dispersas o más precisas según convenga modificando parámetros como el tamaño de la población, la probabilidad de mutación o el método para la fusión de los progenitores.

Finalmente y una vez introducido los conceptos básicos puede presentarse la estructura típica de los algoritmos genéticos que se repite sea cual sea su creador. Se comienza por la generación de la población inicial, comprobando que se no se cumplan las condiciones de parada, y se lleva a cabo la reproducción de los mejores cromosomas teniendo en cuenta las posibles mutaciones y el método de fusión de genes. Tras la nueva generación se vuelve a comprobar las condiciones de parada y si no se cumplen se ejecuta la siguiente iteración repitiendo el mismo proceso. Estas iteraciones no finalizarán hasta cumplir las condiciones de parada. Este proceso se recoge en forma de pseudo-código en el cuadro que sigue.

Algoritmos genéticos

1. Generación aleatoria de la población inicial.
2. Se inicia el **bucle de repetición**:

   Para cada generación:
3. Evaluación de la función objetivo para cada individuo.
4. Selección de los padres
5. Cruce de los padres según los parámetros introducidos.
6. Si se da la condición de mutación para un individuo descendiente, se muta uno de sus genes según los parámetros introducidos para la mutación.
7. Si se cumple la condición de parada, salir del bucle.
8. Se finaliza el **bucle de repetición**.

2.1.2 Población inicial

Gran parte de la responsabilidad de que el algoritmo encuentre la solución óptima del problema que se está resolviendo la tiene la elección de la población inicial. La adecuada selección de esta puede evitar convergencias prematuras a puntos singulares locales o si, por el contrario, se quiere encontrar una solución cercana a unos puntos cuyo resultado es conocido, es posible añadir a la población inicial puntos previamente calculados para que el algoritmo tienda a dicha región del espacio de búsqueda.

El tamaño de la población ha sido uno de los grandes quebraderos de cabezas para los investigadores del campo de la computación evolutiva, ya que una población pequeña puede llevar a resultados insuficientes mientras que una muy grande puede requerir una potencia computacional enorme o, en su defecto, tiempos de computación inasequibles. Algunos investigadores creen que el tamaño de la población debe estar relacionado con el número de variables (N) que presente el problema a resolver, aunque también existe el caso en el que aumentar la población en un problema complicado da lugar a peores soluciones. Por ello, otras personas opinan que la elección de este tamaño se basa en un procedimiento empírico mediante el cual se ejecuta el algoritmo para varios tamaños diferentes y se termina escogiendo aquel que proporciona unos resultados lo suficientemente correctos en un tiempo de ejecución asequible en función de las necesidades.

Una vez se ha conseguido definir la población total, el algoritmo debe generar una población inicial con la que empezará a iterar. Cada individuo (cromosoma) de dicha población inicial es, en sí mismo, una posible solución al problema, por lo que debido a la aleatoriedad en la generación puede darse el caso que algún individuo inicial sea, a su vez, el que proporciona la solución óptima al problema. Para la creación de la población inicial no hay ningún método general si no que cada persona que pretenda crear un nuevo algoritmo puede inventar o adaptar un código para ello.

En este estudio, puesto que se ha utilizado las funciones de MATLAB para la ejecución del algoritmo genético, también se ha utilizado la función de generación de población inicial propia de MATLAB. Dicha función se denomina *gacreationuniform* y recibe como argumentos de entrada la longitud del vector de decisión y las características generales que se le imponen al algoritmo (tamaño de población, posibles individuos iniciales, restricciones de intervalo para cada variable, etc.). Con estas entradas genera una matriz de salida que contiene los individuos de la población inicial obtenidos de forma aleatoria a partir del intervalo de definición de cada una de las variables que conforman el vector de decisión. El núcleo de la generación aleatoria de estos vectores se muestra en el Código 2.1, donde se observa la creación de una variable *span* donde se añade el intervalo factible para cada elemento del vector de decisión y posteriormente se calcula la población como una suma del límite inferior de las restricciones y del producto entre dicho intervalo factible por un número aleatorio entre 0 y 1 generados por la orden *rand*.

Código 2.1 Muestra de la función *gacreationuniform* de MATLAB.

```
.
.
.
totalPopulation = sum(options.PopulationSize);
initPopProvided = size(options.InitialPopulation,1);
individualsToCreate = totalPopulation - initPopProvided;
```

```
if strcmpi(options.PopulationType,'doubleVector')
    % Se inicializa la población a crear
    Population = zeros(totalPopulation,GenomeLength);
    % Se incluye la posible población incial impuesta
    if initPopProvided > 0
        Population(1:initPopProvided,:) = options.InitialPopulation;
    end

    % Creación del resto de población

[SE ESTUDIAN LAS POSIBLES RESTRICCIONES DEL PROBLEMA]

    if ~strcmp(problemtype,'linearconstraints')
        range = options.PopInitRange;
        lowerBound = range(1,:);
        span = range(2,:) - lowerBound;
        Population(initPopProvided+1:end,:) = repmat(lowerBound,
            individualsToCreate,1) + ...
            repmat(span,individualsToCreate,1) .* rand(
                individualsToCreate,GenomeLength);
    else
        Population(initPopProvided+1:end,:) = [];
    end

elseif
 .
 .
 .
```

2.1.3 Selección de los mejores individuos en cada generación

Tras la creación de un conjunto de individuos que conformarán la población incial es necesario un método de selección de individuos que se convertirán en los padres de la nueva generación. Esta selección debe tener relación con el valor asociado de la función objetivo a cada individuo para que así el método escoja como padres a aquellos con mejores resultados, o lo que es lo mismo, a las soluciones más cercanas al óptimo del problema. Para ello se le asocia una probabilidad de selección en función de su valor de la función objetivo permitiendo así que el método se centre en las zonas más prometedoras del espacio de búsqueda. Pese a años de investigación, no se ha logrado encontrar un método general de selección que produzca buenos resultados en cualquier problema a resolver de ahí la existencia de una gran variedad de estos [6]. En lo que sigue se exponen algunos.

RWS (*Roulette Wheel Selection*)

La principal característica de este método es que proporciona a cada individuo la probabilidad de ser escogido según su optimalidad. Si se tiene una población de N individuos, a cada uno le corresponderá la siguiente probabilidad:

$$p(i) = \frac{f(i)}{\sum_{j=1}^{N} f(j)}, \qquad i = 1, ..., N \tag{2.5}$$

Donde $f(i)$ se corresponde con el valor de la función objetivo para un determinado individuo de la población a estudiar. Este método puede implementarse según el siguiente pseudo-código:

Pseudo-código para RWS

1. Se calcula la suma $S = \sum_{j=1}^{N} f(j)$.

Para cada individuo ($1 \leqq i \leqq n$):

2. Generar un número aleatorio (a) entre 0 y S.

3. Inicializar las variables $sum = 0$ y $j = 1$.

4. Se inicia el **bucle de repetición**:

5. Se asigna $sum = sum + f(j)$.

6. Se actualiza el contador $j = j + 1$.

7. Si no se cumple la condición, ($sum < a$ & $j < n$), salir del bucle.

8. Se finaliza el **bucle de repetición**.

9. Se selecciona el individuo j como padre para la siguiente generación.

Una desventaja de este método es que si hay un individuo mucho mejor que el resto tendrá una gran probabilidad de ser escogido y puede ocurrir, incluso, que en cada ejecución del bucle se escoja siempre ese mismo individuo provocando que el algoritmo converja a ese valor pudiendo ser este un mínimo (o máximo) local.

SUS (*Stochastic Universal Sampling*)

Este método se basa en el RWS pero evita el problema de una convergencia prematura hacia mínimos locales. Pseudo-código:

Pseudo-código para SUS

1. Se calcula la media $\bar{f} = \frac{1}{n} \sum_{j=1}^{N} f(j)$.

2. Generar un número aleatorio (a) entre 0 y 1.

3. Inicializar las variables $sum = f(1)$, $j = 0$ y $\Delta = a \cdot \bar{f}$

4. Se inicia el **bucle de repetición**:

5. **Si** $\Delta < sum$:

Selecionar el individuo j

$\Delta = \Delta + sum$

6. **Si no**:

$j = j + 1$

$sum = sum + f(j)$

7. Si no se cumple la condición, ($j < n$), salir del bucle.

8. Se finaliza el **bucle de repetición**.

LRS (*Linear Rank Selection*)

Este es otro de los métodos que surge a partir de la necesidad de mejorar la convergencia prematura del RWS. En este caso se ordenan los individuos según un cierto rango, es decir, si se tiene N

individuos los rangos posibles están comprendidos entre 1 (para el peor) y N (para el mejor). Una vez establecidos los diferentes rangos se prodece a la asignación de sus probabilidades que deben cumplir:

$$p(i) = \frac{rango(i)}{n \cdot (n-1)} \tag{2.6}$$

Conocidas las probabilidades el método de selección puede ejecutarse siguiendo el siguiente pseudo-código:

Pseudo-código para LRS

1. Se calcula $v = \frac{1}{n-2.001}$.
   Para cada individuo $(1 \leqq i \leqq n)$:
2. Generar un número aleatorio (a) entre 0 y v.
3. Se inicia el **bucle de repetición** para $1 \leqq j \leqq n$:
4. **Si** $p(j) \leqq a$:
   Selecionar el individuo j
   Salir del bucle
5. Se finaliza el **bucle de repetición**.

ERS (*Exponential Rank Selection*)

Este método vueve a utilizar la idea del LRS acerca de la realización de una lista ordenada por rangos de los individuos de la población sometida al proceso de selección según su optimalidad. Lo que cambia respecto al LRS es cómo se le asigna un probabilidad a dichos individuos. En el ERS se utiliza se tiene lo siguiente:

$$p(i) = 1.0 \cdot \exp\left(\frac{-rango(i)}{c}\right), \tag{2.7}$$

siendo

$$c = \frac{2n \cdot (n-1)}{6 \cdot (n-1) + n} \tag{2.8}$$

Y, como en el método anterior, una vez conocidas las probabilidades puede realizarse la selección de individuos que viene descrita por el siguiente pseudo-código:

Pseudo-código para ERS

Para cada individuo ($1 \leqq i \leqq n$):

1. Generar un número aleatorio (a) entre $\frac{c}{9}$ y $\frac{2}{c}$.

2. Se inicia el **bucle de repetición** para $1 \leqq j \leqq n$:

3. **Si** $p(j) \leqq a$:

Selecionar el individuo j

Salir del bucle

4. Se finaliza el **bucle de repetición**.

Tras exponer alguno de los métodos posibles de selección de un algoritmo genético se pasa a mostrar el método utilizado en este proyecto para la obtención de resultados. Puesto que se ha utilizado la función de MATLAB del algoritmo genético, va a emplearse también la función de selección *selectionstochunif* incluida en el paquete *Global Optimization Toolbox*. Dicha función implementa el método SUS anterior y para su funcionamiento uno de los argumentos de entrada que debe recibir es el número de padres que se quiera. Tras su ejecución devuelve qué individuos se toman como padres. Parte de dicha función se muestra en el Código 2.2.

Código 2.2 Muestra de la función *selectionstochunif* de MATLAB.

```
.
.
.

expectation = expectation(:,1);
wheel = cumsum(expectation) / nParents;

parents = zeros(1,nParents);

% we will step through the wheel in even steps.
stepSize = 1/nParents;

% we will start at a random position less that one full step
position = rand * stepSize;

% a speed optimization. Position is monotonically rising.
lowest = 1;

for i = 1:nParents % for each parent needed,
    for j = lowest:length(wheel) % find the wheel position
        if(position < wheel(j)) % that this step falls in.
            parents(i) = j;
            lowest = j;
            break;
        end
```

```
    end
    position = position + stepSize; % take the next step.

  .
  .
  .
```

2.1.4 Condición de fusión de los padres

Existen diversas maneras de conseguir fusionar a los cromosomas seleccionados como padres para obtener la nueva generación. A continuación se presentan algunas.

Fusión por un único punto

Es el método más sencillo cuyo funcionamiento se basa en escoger al azar un cierto punto y crear un nuevo individuo con la parte de uno de los padres anterior a dicho punto y con la parte posterior del otro. Una de sus principales desventajas radica en el hecho de que si uno de los padres genera un buen resultado, al partirlo los descendientes no van a heredar esas buenas características directamente.

Fusión por un N puntos

Como su propio nombre indica, este método permite partir los cromosomas progenitores en tantas partes como se pueda y de manera aleatoria se distribuirán entre los descendientes. Este método supone una mejora respecto al de un único punto de corte debido a que hay ocasiones en las que siempre se tienen los mismas partes de cromosomas que se van heredando en las generaciones y no permite al algoritmo recorrer otras zonas del espacio de búsqueda.

Fusión con tres padres

En este caso se produce un cambio respecto a cómo se toman los padres y su fusión, ya que se comienza tomando tres, en lugar de dos, y realiza una comparación entre ellos. El proceso puede exponerse como sigue: se toman dos de ellos y se obtiene si los primeros genes de ambos son similares; si esto se cumple se elige ese gen para la descendencia y si no, se elige el gen correspondiente a la misma posición del tercer padre escogido; repitiendo esto para todos los genes se obtiene un cromosoma descendiente.

Fusión uniforme

Este es el único caso de los expuestos que no realiza una fragmentación de los cromosomas para recombinarlos. El funcionamiento en este caso es algo diferente: parte de la selección de dos de los padres y a medida que recorre la longitud de los cromosomas va obteniendo un valor aleatorio; según sea ese valor escogerá el gen del primero de los padres o del segundo.

Para conseguir una mejor comprensión se va a mostrar parte de la función utilizada durante los cálculos. Esta función, de nuevo, es una función implementada en MATLAB y que utiliza la función del algoritmo genético. Se observa cómo en el Código 2.3 se crea el cromosoma descendiente según un cierto número aleatorio entre 0 y 1 es mayor que 0.5 o no, por tanto, el método que utiliza la función utilizada no es más que el de fusión uniforme.

Código 2.3 Muestra de la función *crossoverscattered* de MATLAB.

```
  .
  .
  .
```

```
[Cálculo previo del número de descendientes]

% Para cada descendiente...
index=1;
for i=1:nKids
    % Se escogen dos padres
    r1 = parents(index);
    index = index + 1;
    r2 = parents(index);
    index = index + 1;
    % De manera aleatoria se va creando el cromosoma descendiente a
        partir de los dos padres
    for j = 1:GenomeLength
        if(rand > 0.5)
            xoverKids(i,j) = thisPopulation(r1,j);
        else
            xoverKids(i,j) = thisPopulation(r2,j);
        end
    end

 .
 .
 .
```

2.1.5 Condición de mutación

Desde el inicio de la investigación en el campo de la computación evolutiva se ha realizado un enfoque más severo sobre los diferentes métodos de fusión de cromosomas dejando en segundo plano los mecanismos de mutación de genes, obviando a veces su importancia en cuanto a que diversifica la población existente y permite al algoritmo explorar el espacio de búsqueda sin tender únicamente a una solución local. Al comienzo se utilizaba una mutación de un gen concreto con una probabilidad pequeña, pero a medida que se fueron realizando nuevos estudios se logró conseguir nuevos métodos como la mutación gaussiana [5], por desplazamiento del marco de lectura (*frame-shift mutation*), por tranlocación, por espejo, etc.

En el trabajo que va a realizarse va a utilizarse la función de MATLAB: *mutationadaptfeasible*, la cual genera mutaciones en los cromosomas de una generación bien con una probabilidad uniforme o bien adaptada, pero siempre cumpliendo con las restricciones impuestas para el vector de decisión. De estos dos casos, se ha hecho uso de una probabilidad fija de mutación que afecta a todos los individuos por igual. Sin embargo, el uso de la mutación adaptada podría haber sido también una buena opción y se planteará en el apartado de trabajos futuros.

Alguna de las ventajas que supone el uso de la mutación adaptada frente a la constante es que permite asociar una probabilidad de mutación a cada individuo con respecto al valor de la función objetivo que generan. Esto ayuda a que cromosomas cercanos al óptimo del problema no muten con tanta facilidad, mientras que los peores de la generación lo hagan para realizar una búsqueda más amplia de la solución objetivo. En el intento de subsanar alguno de los problemas que tiene el fijar una probabilidad fija de mutación, se ha realizado un estudio de dicha probabilidad, o *rate*, de manera que pueda utilizarse para las optimizaciones a estudiar un valor que proporcione un

equilibrio entre las mutaciones de individuos. Sin ser muy alto porque afectaría mucho a los mejores individuos y sin ser muy bajo ya que apenas habría mutación de los individuos malos.

2.1.6 Elitismo

Este concepto permite que el algoritmo no diverja con el paso de las generaciones de una buena solución previamente encontrada. Su implementación consiste en la elección de un cierto porcentaje de la población que posibilita que dicha fracción de población evolucione sin alteraciones a la siguiente generación. Por tanto, es sencillo entender que gracias a que los mejores individuos de cada generación (la élite) pasan de una generación a otra sin modificaciones, el mejor valor que se encuentre en una generación siempre va a estar presente en las siguientes a no ser que se encuentre alguna solución mejor y saque a la anterior de la élite.

El trabajo a realizar sobre el elitismo consistirá en comprobar qué porcentaje de población es el óptimo en lo que respecta al valor de la función objetivo obtenida al final de la ejecución del algoritmo. Nótese que cuanto mayor sea la población élite menor será la posibilidad de generar individuos aleatorios que permitan explorar el espacio de búsqueda, aunque al tener muchos individuos buenos los descendientes serán recombinaciones de estos favoreciendo a la explotación de una cierta zona del espacio en busca de una solución mejor. Si, por el contrario, el porcentaje de elitismo es muy bajo se mantendrán pocos individuos y la búsqueda será más aleatoria.

2.1.7 Condiciones de parada

Con el desarrollo de este tipo de algoritmo, se ha conseguido que su aplicación para la resolución de problemas complejos (problemas de clase NP-completos) con un enorme espacio de búsqueda sea cada vez más frecuente. La propia dificultad de encontrar la solución del problema en un tiempo razonable conlleva a la utilización de una ciertas condiciones de paradas que favorecen la obtención de una solución al problema. La desventaja de ello es clara, si el problema es muy complejo y posee soluciones locales será complicado, si no se escogen bien las condiciones de parada, que la solución resultante sea la óptima y se tendrá una solución local aproximada a esta, que a veces puede ser suficiente.

Las principales condiciones que suelen utilizarse pueden resumirse en lo que sigue:

- Valor límite de la función objetivo: cuando el algoritmo obtenga una solución que tenga como resultado dicho valor se terminarán los cálculos.
- Tolerancia de la función: comprueba que la diferencia entre los resultados de varias generaciones consecutivas no sean menor que una cierta tolerancia dada.
- Generaciones máximas: el algoritmo calculará generaciones de manera aleatoria hasta que se alcance el límite máximo.
- Tiempo máximo: puede establecerse un tiempo fijo de funcionamiento para comprobar la eficiencia del algoritmo o si no se desea que pueda demorarse la obtención de resultados.
- Límite máximo de evaluaciones de la función objetivo

Algunas de las condiciones anteriores se han utilizado durante la resolución de los problemas estudiados, por lo que se especificarán en el apartado de los resultados correspondientes.

2.2 Algoritmo de optimización por enjambre de partículas

Este algoritmo, que tiene su origen en las primeras ideas de J. Kennedy en 1995, fue por primera vez descrito con cierta precisión en un documento también durante 1995 cuando R. C. Eberhart se

unió a Kennedy en el desarrollo e investigación de este método de optimización. Desde el año de su introducción, gran cantidad de investigadores comenzaron a trabajar en las posibles formas de mejorar la idea inicial generando nuevas versiones y aplicaciones que sus creadores iniciales no consiguieron desarrollar.

El funcionamiento de este algoritmo de optimización está basado en el movimiento de un conjunto de animales (una bandada de pájaros, por ejemplo) que tratan de buscar un lugar donde las condiciones para vivir sean las mejores (buen clima, comida, agua, etc.). Por lo general, estos individuos (partículas en términos del algoritmo) parten de una posición inicial desconocida (aleatoria) y a medida que se van moviendo van comprobando si las condiciones en esos nuevos lugares son mejores que las condiciones de las que vienen. En el caso de que se encuentren unas mejores condiciones en un cierto sentido del movimiento, el resto de individuos tienden a moverse en dicha dirección (concepto de *factor de inercia* en el algoritmo). Luego debe existir algún método de transmisión de información entre los individuos y una cierta memoria de cada uno de ellos que permita la comprobación de la calidad de las nuevas condiciones encontradas.

El algoritmo tratará de encontrar el valor óptimo de una cierta función objetivo dada por un problema computacional. Para ello generará una serie de partículas con una posición y una velocidad iniciales totalmente aleatorias aunque deben estar contenidas dentro del espacio de búsqueda limitado por las restricciones propias del problema que se esté resolviendo. Cada una de estas partículas será un conjunto de variables definidas por el vector de decisión y que, en su evaluación, genera un valor de la función objetivo. A partir de dichos valores, el algoritmo tiene la capacidad de comprobar cuál es el mejor y provocar que la velocidad de movimiento de las partículas cambien un poco hacia dicho punto. Conforme pasan las iteraciones, las partículas habrán ido avanzando hacia un punto óptimo o no, puesto que pueden existir mínimos locales en la función que se evalúa, y se finalizará la simulación cuando se produzcan alguna de las condiciones de parada que pueden imponerse.

Al igual que el algoritmo genético, esta optimización de partículas es aplicable a cualquier tipo de función sin influir sus características (continuidad, derivabilidad, etc.) por lo que se convierte también en un método de optimización robusto de gran aplicabilidad, de ahí su amplio uso. En los siguientes apartados va describirse con un mayor rigor y detalle el funcionamiento del algoritmo.

2.2.1 Conceptos básicos y estructura

Previamente a la aplicación del algoritmo a los problemas tratados en el trabajo van a exponerse los conceptos básicos que permiten un buen entendimiento del mismo. Entre ellos se encuentran:

- La función objetivo y el vector de decisión
- La generación inicial de partículas
- La interacción entre partículas y su evolución

Utilizando la misma notación que en el algoritmo genético, la función a optimizar, $J(x)$, se le denomina función objetivo. Para ello, se genera el conjunto de partículas que evaluarán la función en diferentes puntos, pero siempre dentro de un cierto espacio de búsqueda (χ) donde la función está definida. Por lo general, la solución óptima suele producirse en la minimización de la función objetivo

$$\min_{x \in \chi} \; J(x), \tag{2.9}$$

aunque hay casos en los que se desea maximizar la función. Dicha maximización se consigue cambiándole el signo y, ahora sí, aplicando la minimización

$$\max_{x\in\chi} J(x) = \min_{x\in\chi} -J(x), \tag{2.10}$$

donde x es el vector de decisión, compuesto por un conjunto de valores asociados a las variables respecto a las cuales se está realizando la optimización y que lleva asociado un único valor de esta función.

En relación a la generación de las partículas encargadas de la búsqueda del óptimo existen diferentes métodos, aunque que todos parten de una generación aleatoria cumpliendo con las restricciones impuestas. Uno de los parámetros que caracterizan dicha población inicial será el número total de partículas utilizadas. Cuanto mayor sea dicho número más sencillo será encontrar la solución buscada de la función, ya que se abarca un espacio mayor de búsqueda al mismo tiempo. Sin embargo, como consecuencia, el tiempo de ejecución aumentará al ser más pesado el cálculo de la evolución de una gran cantidad de partículas. Puede verse como en el Capítulo 5 se realiza un estudio, previo a la aplicación del algoritmo al problema a resolver, de comparación "Tamaño enjambre - Tiempos de ejecución" y se escoge un valor adecuado. Para un problema de dimensión N, cada partícula estará definida por un vector del tipo:

$$x = [p_1, p_2, ..., p_N] \tag{2.11}$$

siendo p_i cada una de las variables de las que dependerá la optimización.

Es en la forma de interacción entre partículas y el método utilizado para la evolución de la posición de estas donde este tipo de algoritmo ha sido muy investigado desde su presentación al campo de la optimización. El uso de una sola partícula para pretender encontrar un cierto valor de una función es inviable de ahí el uso de un conjunto de ellas (un *enjambre*). Al usar un enjambre de partículas debe establecerse algún modo de relacionarlas, porque si no se comportarán como lo haría una partícula por sí sola. Esta relación se consigue a través de unos parámetros que permiten ajustar el peso que se le da al mejor punto encontrado por la propia partícula y al mejor punto encontrado por el resto de partículas.

Si se escoge un valor alto del parámetro que define la importancia del mejor resultado encontrado por cada partícula y un valor bajo al que define la importancia del mejor resultado global, se induce una situación en la que cada partícula tiende siempre hacia el punto donde encontró su mejor resultado sin apenas desviaciones. Esta configuración suele utilizarse una vez se ha encontrado un buen resultado de la función objetivo y se quiere rastrear una cierta zona con mayor precisión. En el caso opuesto, si se le da mayor importancia al mejor resultado encontrado por cualquiera de las partículas, se consigue un escrutinio más completo de todo el espacio de búsqueda. Por lo tanto, la situación ideal sería comenzar con una búsqueda general y una vez encontrado los mejores resultados particularizar la búsqueda a dichas zonas en concreto.

Junto a estos parámetros de interrelación entre partículas se tiene otro parámetro que proporciona el sentido físico al algoritmo y que alude a la inercia de la propia partícula. Es decir, si hay una partícula moviéndose en una cierta dirección pero se encuentra una solución muy cercana a la óptima justo en ese instante y en el sentido opuesto, la partícula no cambia su sentido de desplazamiento instantáneamente si no que se verá forzada a un proceso de frenada y aceleración en el sentido opuesto como si de un objeto real en movimiento se tratase.

Finalmente, para realizar la siguiente iteración del algoritmo se tiene que actualizar la posición de cada una de las partículas según la velocidad resultante de sumar el término de inercia, el término debido al mejor resultado individual y el término debido al mejor resultado global encontrado por el resto del enjambre. La ejecución del algoritmo finalizará al cumplirse alguna de las condiciones de parada como puede verse en el pseudo-código expuesto.

Algoritmo de optimización por enjambre de partículas [10]

1. Generación aleatoria de las posiciones y velocidades iniciales de las partículas según las restricciones del espacio de búsqueda.
2. Se inicia el **bucle de repetición**:

   Para cada partícula:
3. Evaluación de la función objetivo.
4. Comparar el nuevos valor obtenido con el mejor encontrado hasta el momento. Actualizar el mejor valor personal.
5. Comparar los mejores valores obtenidos por todas las partículas y actualizar el mejor valor global.
6. Actualizar la velocidad teniendo en cuenta la inercia, la atracción al mejor personal y la atracción al mejor global.
7. Si se cumple la condición de parada, salir del bucle.
8. Se finaliza el **bucle de repetición**.

2.2.2 Elecciones iniciales: posición y velocidad.

Al igual que en el algoritmo genético, la elección de unas buenas posiciones y velocidades para las partículas del enjambre puede: llevar a encontrar la solución óptima en un tiempo razonable; provocar que las partículas tiendan a un mínimo local; o que no se llegue nunca a la solución buscada. Las características que pueden modificarse del enjambre, y que ayudan a mejorar la actuación del algoritmo, son: el número total de partículas (tamaño del enjambre) y el uso de un enjambre inicial.

Comenzando por el tamaño del enjambre, no ha habido grandes investigaciones acerca del tamaño óptimo si no que los científicos, que han utilizado este tipo de algoritmo en sus estudios, seguían las recomendaciones iniciales del 1995 donde Kennedy expuso que se debía restringir el tamaño a unas 20 o 50 partículas. Sin embargo, en un estudio reciente de mayo de 2020 por los investigadores polacos A. P. Piotrowski, J. J. Napiorkowski y A. E. Piotrowska [9] se comprueba para una gran diversidad de problemas y variantes del algoritmo de optimización por enjambre de partículas que el tamaño óptimo del enjambre suele estar entre 70 y 500 partículas, luego concluyeron que el tamaño propuesto por Kennedy era pequeño. También aclaran que en el caso de tener una función objetivo muy compleja puede ser interesante aumentar más el tamaño del enjambre, por lo que se comprobará por medios propios cuál es el tamaño adecuado para los problemas que se estudian en el trabajo.

Adicionalmente, es posible introducir un enjambre inicial que se puede haber obtenido por otros métodos. El hecho de fijar las posiciones de una serie de partículas favorece a que el movimiento de dichas partículas sea por una zona del espacio a estudiar cercana a ese punto inicial. Esto suele utilizarse cuando se cree haber obtenido una solución cercana a la óptima y se desea focalizar la

búsqueda a una zona más concreta. Uno de los métodos para la obtención de una población inicial puede ser la aplicación de otro algoritmo de optimización y, por ello, va a introducirse en el estudio de los problemas un caso en el que se concatenan ejecuciones del algoritmo genético con el de optimización de partículas traspasándose los resultados en forma de poblaciones iniciales.

Como se ha hecho para el algoritmo genético, se ha utilizado una función propia de MATLAB que implementa el algoritmo de optimización: *particleswarm*. Para el correcto funcionamiento de la misma, habrá que darle como entradas el problema a resolver, el número de variables de optimización, las restricciones a dichas variables y una serie de opciones donde se incluyen los parámetros comentados hasta el momento: tamaño de población, coeficiente de inercia, etc. Además, como opción adicional puede escogerse la función deseada para la generación de las posiciones iniciales de las partículas del enjambre. La función utilizada se denomina *pswcreationuniform*.

Para la creación de las posiciones iniciales, la función debe recibir el número de variables por las que estará definida cada partícula y las restricciones impuestas por el problema. También, si se le proporciona a la entrada un cierto número de partículas iniciales las tendrá en cuenta cuando genere la población total. La función calcula cuál es el intervalo viable de cambio de las variables para quedar dentro de los límites a través del parámetro *span* y posteriormente suma al límite inferior dicho intervalo multiplicado por un número aleatorio entre 0 y 1 generados con la orden *rand*.

Código 2.4 Muestra de la función *pswcreationuniform* de MATLAB.

```
.
.
.
% Obtención del tamaño del enjambre total e inicial
numParticles = options.SwarmSize;
numInitPositions = size(options.InitialSwarm, 1);
numPositionsToCreate = numParticles - numInitPositions;

% Inicialización del enjambre
swarm = zeros(numParticles,nvars);

% Uso de las posiciones iniciales (si las hubiera)
if numInitPositions > 0
    swarm(1:numInitPositions,:) = options.InitialSwarm;
end

% Creación del resto de posiciones según las restricciones y de manera
    aleatoria
span = ub - lb;
swarm(numInitPositions+1:end,:) = repmat(lb,numPositionsToCreate,1) +
    repmat(span,numPositionsToCreate,1) .* rand(numPositionsToCreate,
    nvars);
 .
 .
 .
```

Una vez obtenidas las posiciones iniciales para cada una de las partículas que conformen el enjambre habrá que asignarles una velocidad inicial. Esta atribución viene implementada en la subfunción *makeState* dentro de la propia función *particleswarm* de MATLAB. Mediante un proceso parecido al de la elección de las posiciones, en este caso se calcula la variación máxima que puede sufrir una partícula en el paso de una iteración del algoritmo para luego obtener la velocidad real para cada una de ellas. Véase en el Código 2.5 como, generalmente, la variable *vmax* va a coincidir con la variable *span* utilizada en el cálculo de las posiciones.

Código 2.5 Muestra de la subfunción *makeState* (de *particleswarm*) de MATLAB.

```
.
.
.
vmax = min(ub-lb, options.InitialSwarmSpan);
state.Velocities = repmat(-vmax,numParticles,1) + ...
    repmat(2*vmax,numParticles,1) .* rand(numParticles,nvars);
 .
 .
 .
```

2.2.3 Interacción entre partículas y evolución del algoritmo

Ya se conoce que cada partícula tendrá asociada una cierta posición (una solución) del espacio de búsqueda y una velocidad que permitirá ir actualizando su posición con el paso de las iteraciones. En este apartado van a comentarse los diferentes factores a tener en cuenta en la actualización de la velocidad y, por ende, en la posición.

Antes de especificar las características de dichos factores va a presentarse cómo se consigue la evolución del algoritmo. Una vez se ha evaluado la función objetivo para todas las partículas y se han actualizado los mejores valores personales de cada una de ellas y el mejor valor global, se debe actualizar tanto la posición como la velocidad de cada partícula. Para ello, las ecuaciones que se utilizan son [1]:

$$pos_i(t+1) = pos_i(t) + v_i(t); \tag{2.12}$$

$$v_i(t+1) = \omega \cdot v_i(t) + c_1 \cdot r_1 \cdot (p_i^{mejor} - p_i(t)) + c_2 \cdot r_2 \cdot (p_g^{mejor} - p_i(t)); \tag{2.13}$$

donde $pos_i(t)$ indica la posición de cada partícula i en la iteración t, $v_i(t)$ la velocidad de cada partícula i en la iteración t, ω el factor de inercia, r_1 y r_2 son dos parámetros aleatorios del intervalo $[0,1]$, c_1 y c_2 son las constantes de atracción al mejor personal y global respectivamente, p_i^{mejor} el mejor valor encontrado por la partícula i y, finalmente, p_g^{mejor} el mejor valor encontrado por el enjambre en su conjunto. Durante la aplicación de las ecuaciones (2.12) y (2.13), el parámetro i variará entre 1 y el número de partículas del enjambre, mientras que el parámetro t se incrementará una unidad por cada iteración realizada hasta alcanzar un máximo que debe ser impuesto por el programador.

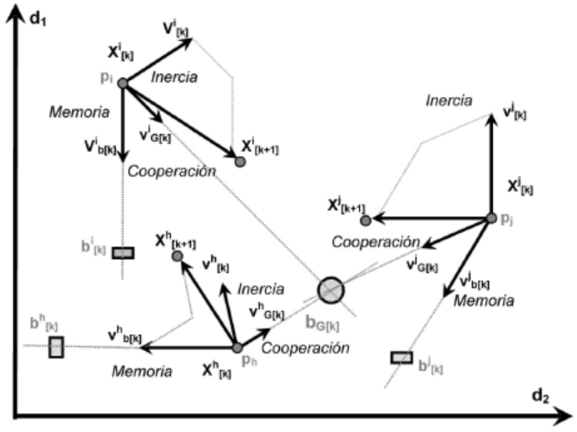


Figura 2.1 Representación de los diferentes factores que afectan en la actualización de la posición de una partícula. (Imagen extraída de [2]).

Factor de inercia

Este factor fue introducido por Shi y Eberhart [12] en 1998 con la intención de mejorar el control de la exploración y explotación, ya que determina la contribución de la velocidad de una partícula a la nueva que se calcule en la siguiente iteración. Puede verse en la Ecuación (2.13) que cuanto mayor sea el factor ω mayores serán los cambios en la velocidad en cada iteración y mayores serán las diferencias entre dos posiciones de partícula consecutivas. Este caso mejoraría la exploración. En cambio, si lo que se desea es un estudio concentrado entorno a una cierta posición conviene definir ω con un valor bajo favoreciendo, por tanto, la explotación.

Pese a que en su presentación de 1998 este factor se mantuvo constante con las iteraciones, con las nuevas investigaciones han surgido nuevas estrategias: *Random Inertia Weight* [3], *Linearly Decreasing strategy* [16], *Adaptive Inertia Weight* [7], etc. Para el trabajo que se presenta se ha optado por mantener el factor de inercia constante, aunque la función de MATLAB utilizada da la posibilidad de hacer uso del caso de factor de inercia adaptado lo cual será incluido como una mejora al estudio realizado en el Capítulo 7.

Constantes de atracción al mejor valor

Acompañando al factor de inercia, se encuentran las constantes c_1 y c_2 que establecen la influencia del mejor valor personal y global respectivamente a la nueva velocidad que se calcula en cada iteración (véase la Ecuación (2.13)). Se observa que cuanto mayor sea la distancia entre la mejor posición personal (o global) y la posición actual mayor será la contribución de dicho término. Además de escalar dicha contribución con las constantes c_1 y c_2, se utilizan las variables aleatorias r_1 y r_2 para dotar de una cierta aleatoriedad al proceso de actualización de la velocidad.

Respecto a las constantes de atracción se tienen dos casos especiales, que serán estudiados en el Capítulo 5:

- Modelo de atracción personal, donde se elimina la componente social ($c_2 = 0$).
- Modelo de atracción social, que excluye la componente personal ($c_1 = 0$).

Para la elección adecuada de estos parámetros a la hora de optimizar, se ha hecho un análisis denso de diferentes configuraciones hasta escoger la más adecuada según se aplique a un problema u otro. Como se verá en el Capítulo 5, se han estudiado los parámetros de forma que si se realiza una suma ponderada dan lugar a un cierto valor que permite entender cómo de rápido o lento se mueven las partículas del enjambre.

2.2.4 Condiciones de parada

Como cualquier algoritmo de optimización, el algoritmo de optimización por enjambre de partículas también requiere de una serie de condiciones que eviten que una ejecución del mismo se alargue hasta la eternidad. Al imponer las condiciones de parada, quizás se esté evitando que el algoritmo encuentre el óptimo de un cierto problema pero es importante que los tiempos que tarda en encontrar un resultado cercano al punto óptimo (o el propio óptimo) sean razonables.

Para ello, la función de MATLAB implementa las típicas condiciones de parada que se utilizan en este algoritmo:

- Valor límite de la función objetivo: el algoritmo se ejecuta hasta encontrar dicho valor.
- Iteraciones máximas sin mejora: esta condición combinada con un valor de tolerancia de la función que se proporciona por el programador, permite al algoritmo parar en el caso alcanzar dicho número de iteraciones en el que el resultado obtenido no varía más de la tolerancia comentada.
- Iteraciones máximas: se refiere al número total de actualizaciones de posición de las partículas que va a realizar el algoritmo.
- Tiempo máximo: tiempo que durará la ejecución del algoritmo si no se ha parado anteriormente por otra de las condiciones.

En la presentación de los resultados, se irán comentando cuáles han sido las condiciones escogidas para cada uno de los problemas que se optimizan.

3 Conceptos de la mecánica orbital

Teniendo presente que el objetivo principal del trabajo es comprobar la optimización de misiones interplanetarias con algoritmos metahurísticos parece lógico dedicar un capítulo a la introducción de los conceptos básicos de la mecánica orbital utilizados en los códigos de modelización de los problemas. La mecánica orbital debe su origen a aquellas personas curiosas de las antiguas civilizaciones (egipcia, maya, etc.) que se preguntaban qué era lo que veían en el cielo y cómo se comportaban lo que se conoce actualmente como estrellas o planetas. En dicha época, estos cuerpos que se mueven por el espacio eran indistinguibles entre ellos y durante años se buscó la forma de entender su comportamiento, surgiendo así la Astronomía que resultaría con el paso de los años en la Mecánica orbital. Conocidos filósofos (y científicos experimentales por pasión), como Aristóteles, ya plantearon un modelo geocéntrico que explicaba aproximadamente lo que se observaba en el cielo, pero debido a algunas incoherencias con las observaciones, otros científicos más modernos comenzando por Copérnico y siguiendo con Galileo, Kepler o Newton, se acabó imponiendo un sistema heliocéntrico, que ha sido demostrado gracias a las diferentes misiones interplanetarias realizadas por el avance tecnológico en la sociedad.

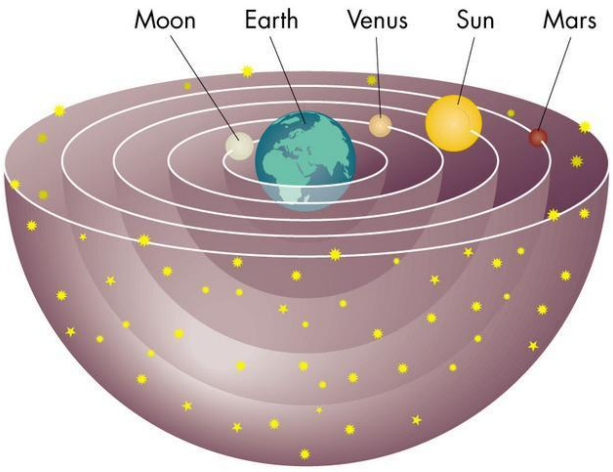


Figura 3.1 Esquema sencillo del modelo geocéntrico planteado.

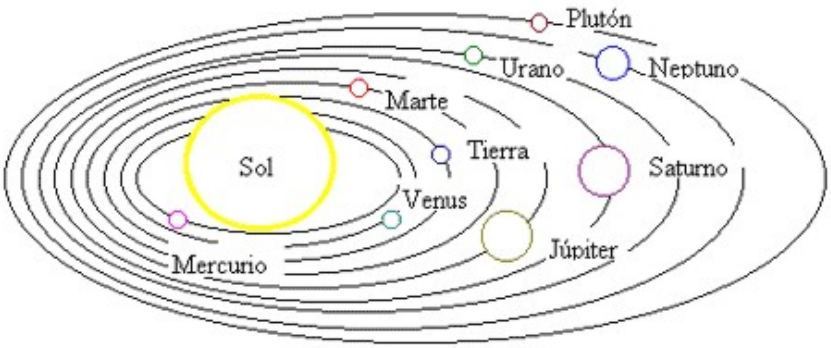


Figura 3.2 Esquema sencillo del modelo heliocéntrico actual. Imagen extraída de [8].

Puesto que la idea de conocer a fondo el espacio que nos rodea siempre ha estado presente a lo largo de los años, con la llegada del siglo XX y una vez la tecnología alcanzó un nivel de desarrollo adecuado comenzó la era espacial. Los primeros países que lograron fabricar cohetes que consiguieran mandar un instrumento al espacio fueron: Estados Unidos, Rusia (URSS) y Alemania. El primer lanzamiento fue ruso, que puso en órbita al satélite Sputnik I, Alemania se centró principalmente en la fabricación de misiles y Estados Unidos logró mandar algunos satélites para, finalmente, cerrar esta primera etapa de era espacial con la llegada a la Luna por parte de la misión Apollo 11.

Figura 3.3 Astronautas de la misión Apollo 11: N. Armstrong, M. Collins y E. "Buzz" Aldrin.

Como ya se conoce la era espacial ha seguido desarrollándose desde la llegada a la Luna y se han llevado a cabo misiones como: Galileo (Júpiter), Cassini/Huygens (Saturno), New Horizons (Plutón), Rosetta (cometa 67P/Churyumov-Gerasimenko), etc. Algunas de estas misiones son las que se pretenden estudiar en este trabajo realizando optimizaciones por algoritmos metaheurísticos comentados en el Capitulo 2 con el fin de comprobar la aplicabilidad de los mismos a problemas reales obteniendo buenos resultados para unos tiempos de ejecución viables. Previo a mostrar los problemas que se han estudiado y los resultados obtenidos hay que exponer algunos conceptos básicos de la mecánica orbital que son importantes para entender los modelos utilizados en las optimizaciones.

Figura 3.4 Imagen conceptual de la sonda Galileo orbitando Júpiter.

3.1 Leyes horarias. Teorema de Lambert

Las leyes horarias son la herramienta que permiten estudiar las situaciones en las que un cuerpo está orbitando a otro mucho más masivo (posición y tiempos de vuelo). En este apartado se pretende desarrollar brevemente el proceso de obtención de la ecuación general de las cónicas ya que va a ser de gran utilidad para entender algunos desarrollos posteriores del trabajo, exponer los diferentes tipos de órbitas que pueden encontrarse en la realidad junto a sus leyes horarias y, por último, una explicación básica del teorema de Lambert y cómo se ha aplicado en el modelo utilizado.

3.1.1 Ecuación general de las cónicas

Para la obtención de dicha ecuación se parte de una serie de hipótesis que facilitan el proceso, como suponer que se tiene un sistema de dos cuerpos aislado del exterior con un sistema de referencia inercial, donde ambas masas se consideran puntuales. Si además se considera que una de las dos masas es mucho mayor que la otra, puede demostrarse que dicha masa permanecería prácticamente estática mientras la otra orbitaría alrededor siguiendo la ecuación del movimiento:

$$\ddot{\mathbf{r}} = -\mu \frac{\mathbf{r}}{r^3} \tag{3.1}$$

Si se quiere llegar a la ecuación general de las cónicas hay que resolver dicha ecuación diferencial que tendrá como condiciones iniciales los seis elementos orbitales que se estudiarán en el siguiente apartado. Respecto a la resolución de la ecuación habrá que utilizar las conocidas *cantidades conservadas*, que no son más que una serie de magnitudes que, debido a las hipótesis consideradas, se mantienen constante durante el movimiento orbital. Estas *cantidades conservadas* son:

- **Energía específica**: Si en la ecuación del movimiento (3.1) multiplicamos escalarmente a ambos lados por $\dot{\mathbf{r}}$ y se desarrollan los productos se obtiene la primera *cantidad conservada*

$$\frac{d}{dt}\left(\frac{v^2}{2}\right) = \frac{d}{dt}\left(\frac{\mu}{r}\right) \quad \longrightarrow \quad \frac{v^2}{2} - \frac{\mu}{r} = cte = \varepsilon \tag{3.2}$$

- **Momento cinético**: Si se quiere hacer aparecer el momento cinético en la ecuación del movimiento hay que multiplicar vectorialmente por la izquierda en ambos términos de la

ecuación (3.1). Tras los desarrollos necesarios se llega a:

$$\frac{d}{dt}(\mathbf{r}\times\dot{\mathbf{r}}) = \mathbf{0} \quad \longrightarrow \quad \mathbf{r}\times\mathbf{v} = \mathbf{cte} = \mathbf{h} \tag{3.3}$$

- **Vector excentricidad**: En este tercer caso se llega a la *cantidad conservada* si se realiza un producto vectorial por la derecha en la ecuación (3.1) con el momento cinético (**h**). Operando se obtiene:

$$\frac{d}{dt}(\mathbf{v}\times\mathbf{h}) = \frac{d}{dt}\left(\frac{\mu\mathbf{r}}{r}\right) \quad \longrightarrow \quad \mathbf{v}\times\mathbf{h} - \frac{\mu\mathbf{r}}{r} = \mathbf{cte} = \mu\mathbf{e} \tag{3.4}$$

Tras la obtención de dichas magnitudes, especialmente la tercera, se está en disposición de calcular la ecuación que se busca. Para ello basta con realizar un producto escalar en la ecuación (3.4), de donde se obtendrá

$$h^2 - \mu r = \mu e r\cos\theta, \tag{3.5}$$

y si se despeja r, la ecuación general de las cónicas:

$$r = \frac{p}{1+e\cos\theta} \tag{3.6}$$

siendo $p = h^2/\mu$ el *parámetro* de la cónica.

3.1.2 Tipos de cónicas y leyes horarias

A partir de la ecuación (3.6) puede obtenerse cualquier tipo de cónica simplemente variando el semieje mayor, a, y la excentricidad, e:

- **Elipse**: En este caso el semieje mayor debe ser positivo puesto que el parámetro p también es positivo y e debe estar en el intervalo $[0,1)$. Cumpliendo así la ecuación:

$$p = a(1-e^2) \tag{3.7}$$

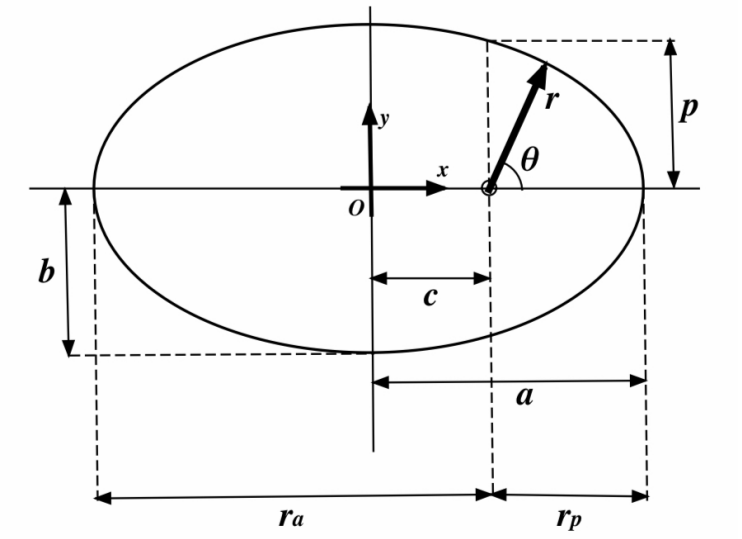


Figura 3.5 Caracterización de la elipse. Extraída de [14].

Además, pueden calcularse algunos valores sencillos como radio de periapsis o apoapsis en función de los elementos orbitales de la ecuación (3.6). El caso de e=0 proporciona la situación en la que la elipse degenera a una circunferencia.

- **Parábola**: Esta es la situación límite que separa las elipses de las hipérbolas y es que para tener una parábola la excentricidad debe valer 1. Si se estudia la ecuación de la cónica se observa como el punto más cercano al foco en este caso estará a

$$r_p = \frac{p}{2} \tag{3.8}$$

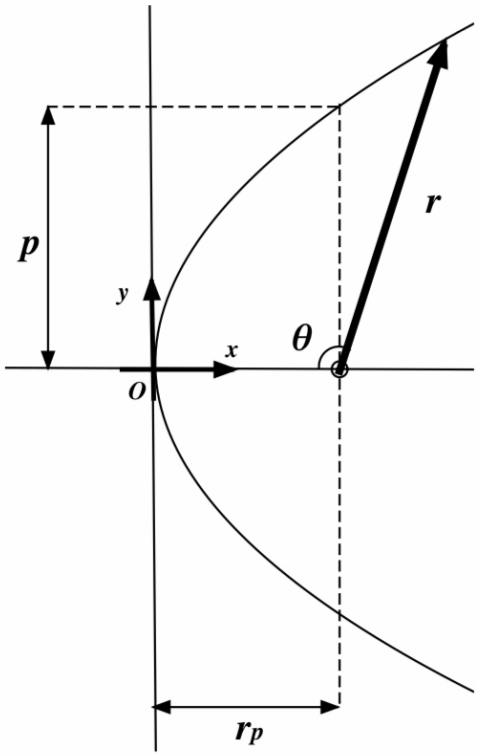


Figura 3.6 Caracterización de la parábola. Extraída de [14].

- **Hipérbola**: Para los valores restantes de la excentricidad, es decir, los del intervalo $(1, +\infty)$ se obtienen hipérbolas. En este caso el semieje mayor será negativo puesto que se debe cumplir

$$p = a(1 - e^2), \tag{3.9}$$

donde p sigue siendo positivo. Para la hipérbola también puede obtenerse de manera sencilla la distancia al punto más cercano al foco (periapsis) que será

$$p = a(1 - e), \tag{3.10}$$

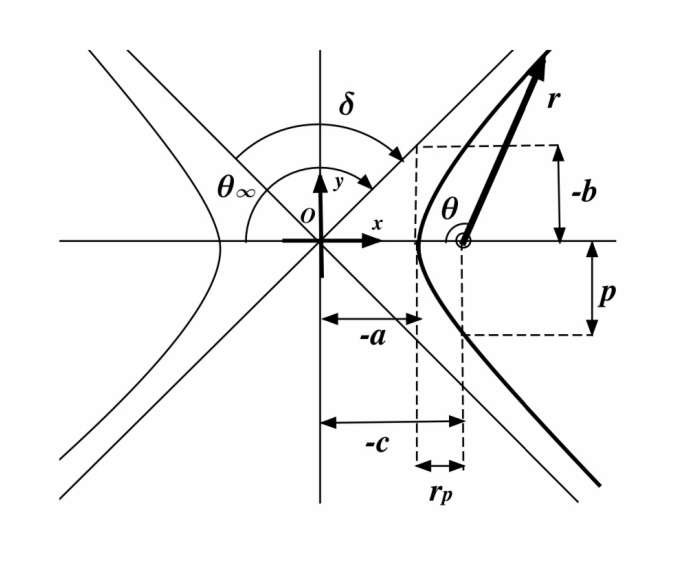


Figura 3.7 Caracterización de la hipérbola. Extraída de [14].

En cuanto a las leyes horarias, permiten localizar un cuerpo en una cierta órbita para un instante determinado o, su caso inverso, dada una órbita y un punto inicial y final calcular el tiempo que emplea dicho cuerpo para realizar el recorrido. Puesto que no se van a utilizar explícitamente en el modelo usado se exponen brevemente las ecuaciones para tener un mayor conocimiento a la hora de entender el teorema de Lambert, si se quisiera profundizar en los cálculos geométricos que proporcionan las ecuaciones véase los apuntes de Rafael Vázquez [14]. Estas ecuaciones dependerán de la órbita a tratar:

- **Órbita elíptica**:

  1.
  $$\tan\frac{\theta}{2} = \sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2} \tag{3.11}$$

  2.
  $$M = E - e\sin E \qquad \text{(Ecuación de Kepler)} \tag{3.12}$$

  3.
  $$M = n\Delta t \tag{3.13}$$

  Aclarar que E se denomina anomalía excéntrica y se define geométricamente (véase la Figura 3.8), M es la anomalía media, que se refiere al ángulo que recorrería el cuerpo si la órbita fuera circular y $n = \sqrt{\mu/a^3}$ es la velocidad orbital media del cuerpo. En la definición geométrica de E entra en juego la igualdad:

  $$\frac{y_C}{y_E} = \frac{a}{b} \tag{3.14}$$

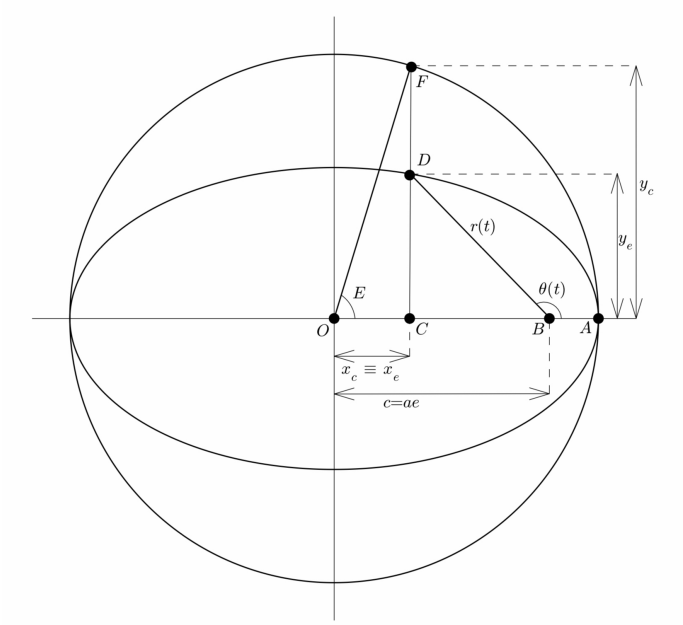


Figura 3.8 Obtención geométrica de la anomalía excéntrica. Extraída de [14].

- **Órbita parabólica**:

  1.
  $$B = 3\sqrt{\frac{\mu}{p^3}}\Delta t \tag{3.15}$$

  2.
  $$2B = 3\tan\frac{\theta}{2} + \tan^3\frac{\theta}{2} \qquad \text{(Ecuación de Barker)} \tag{3.16}$$

  3.
  $$\tan\frac{\theta}{2} = z - \frac{1}{z}, \quad \text{donde} \quad z = \sqrt[3]{B + \sqrt{B^2+1}} \tag{3.17}$$

  En este caso B no es más que un simple nexo de unión entre las ecuaciones y no tienen ningún sentido real por sí mismo.

- **Órbita hiperbólica**:

  1.
  $$\tan\frac{\theta}{2} = \sqrt{\frac{1+e}{e-1}}\tanh\frac{H}{2} \tag{3.18}$$

  2.
  $$N = e\sinh H - H \qquad \text{(Ecuación de Kepler hiperbólica)} \tag{3.19}$$

  3.
  $$N = n\Delta t \tag{3.20}$$

  Para la hipérbola, H sí que tiene sentido y se refiere al ángulo hipérbolico obtenido geométricamente (véase la Figura 3.9) de nuevo teniendo en cuenta que se debe cumplir la relación:

  $$\frac{y_H}{y_E} = \frac{a}{b} \tag{3.21}$$

Por otro lado, N se denomina anomalía media hiperbólica y la velocidad media orbital sería entonces $n = \sqrt{-\mu/a^3}$.

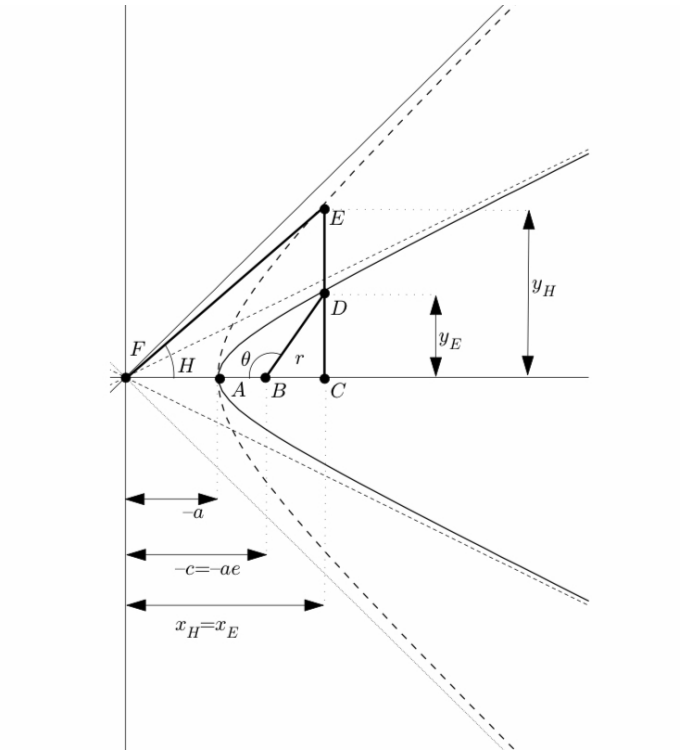


Figura 3.9 Obtención geométrica del ángulo hiperbólico. Extraída de [14].

Tanto los tiempos calculados como la anomalía verdadera, θ, están referidos al periapsis, luego para el cálculo del tiempo empleado en viajar entre dos puntos habrá que hacer uso de un incremento.

En cuanto al método de resolución para el caso elíptico e hiperbólico, si el dato principal es la anomalía verdadera hay que seguir el orden 1-2-3 de las ecuaciones para obtener el tiempo, pero si, por el contrario, el dato es temporal habrá que utilizar las ecuaciones en el orden 3-2-1 para obtener la posición en dicho instante. En cambio, en el caso parabólico si el dato es el tiempo se resuelven las ecuaciones en el orden 1-2-3 para obtener la posición (θ) y en el caso opuesto en el orden 2-1.

3.1.3 Teorema de Lambert. Aplicación al trabajo

El teorema de Lambert puede aplicarse a cualquiera de los tres tipos de órbitas estudiadas, pero se va centrar la explicación para el caso de una órbita elíptica que son las utilizadas para las transferencias entre las órbitas de planetas de los problemas que se van a optimizar. El teorema expone que el tiempo de vuelo entre dos puntos de una elipse, r_0 y r_1, depende únicamente de la cuerda, c, de la suma $s = r_0 + r_1$ y del semieje mayor de la elipse, a.

Para el cálculo del tiempo de vuelo se comienza obteniendo los ángulos α y β con las ecuaciones:

$$\cos\alpha = 1 - \frac{s+c}{2a} \tag{3.22}$$

$$\cos\beta = 1 - \frac{s-c}{2a} \tag{3.23}$$

Conocidos α y β, el tiempo de vuelo se define:

$$t_v = \frac{(\alpha - \sin\alpha) - (\beta - \sin\beta)}{n} \tag{3.24}$$

Puesto que en los problemas que van a optimizarse el dato de entrada es el propio tiempo de vuelo, se aplicará el teorema de Lambert de manera inversa y se conseguirá calcular los parámetros de la órbita de transferencia que una dos puntos fijados de antemano en dicho tiempo dado. Este método inverso es lo que se conoce como problema de Lambert.

3.2 Elementos orbitales

Los elementos orbitales de la órbita de un cuerpo en el espacio son un conjunto de seis parámetros que permiten definir su órbita alrededor de cualquier otro cuerpo de forma totalmente unívoca. Estos elementos, que surgen del problema de los dos cuerpos sin perturbaciones, son los siguientes:

- Ascensión recta del nodo ascendente, Ω (RAAN): Indica la posición del nodo ascendente de la órbita medido respecto a una dirección de referencia (generalmente el Primer punto de Aries, γ).
- Argumento de periapsis, ω: Indica la posición del periapsis de la órbita medida desde el nodo ascendente en el plano orbital. Orienta cómo de rotada está la órbita en el plano.
- Inclinación de la órbita, i: Ángulo que forma el plano orbital con el plano de referencia.
- Semieje mayor de la órbita (elíptica), a: Permite determinar el tipo de órbita.
- Excentricidad, e: Permite calcular la forma de la órbita.
- Anomalía verdadera, θ: Determina la posición del cuerpo que orbita en cualquier instante de tiempo.

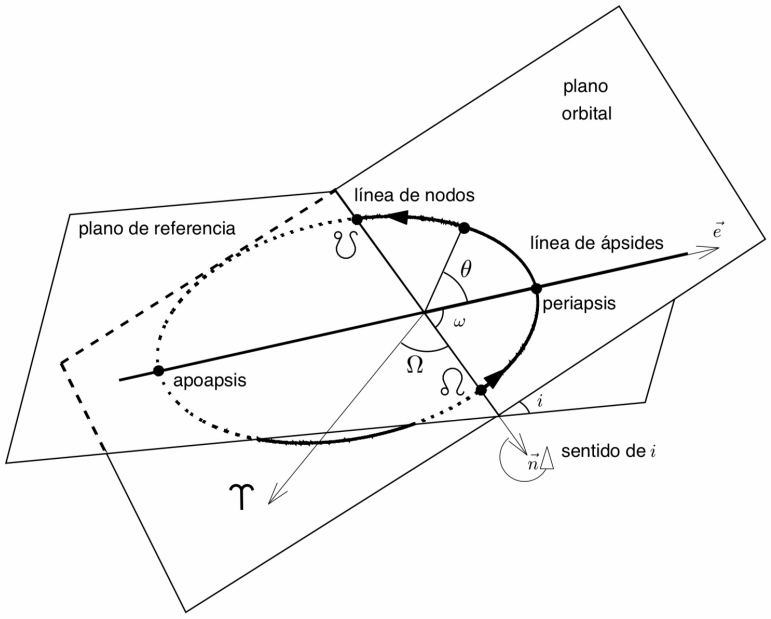


Figura 3.10 Esquema que representa los diferentes elementos orbitales. Extraído de los apuntes de Rafael Vazquez [14].

Añadir que según los valores de estos elementos orbitales va a tenerse un tipo de órbita u otra entre las cuales pueden encontrarse varios casos degenerados. Por ejemplo, para órbitas entorno a la Tierra, según el valor de la inclinación pueden tenerse órbitas ecuatoriales si $i = 0$, órbitas polares si $i = 90$ o retrógradas si $i > 90$. Respecto a los casos degenerados pueden encontrarse los siguientes:

- Órbitas elípticas ecuatoriales: No están bien definidos los parámetros Ω y ω.
- Órbitas circulares no ecuatoriales: No están bien definidos ω y θ.
- Órbitas circulares ecuatoriales: No están bien definidos Ω, ω y θ.

A continuación se presentan brevemente los dos métodos de conversión entre la posición y velocidad de un cuerpo en una cierta órbita y sus elementos orbitales.

3.2.1 Conversión de posición y velocidad a elementos orbitales

En este caso, son datos tanto el vector posición, **r**, como el vector velocidad, **v**, cuyos módulos van a escribirse como r y v. También es dato el parámetro μ asociado al cuerpo masivo entorno al cual se orbita.

Comenzando por el cálculo del semieje mayor de la órbita, habría que despejar el parámetro de la ecuación de la energía específica, igualando la obtenida en la Sección 3.6 con la proporcionada por la ecuación de las fuerzas vivas:

$$\varepsilon = \frac{v^2}{2} - \frac{\mu}{r} = -\frac{\mu}{2a} \tag{3.25}$$

En el caso de la excentricidad, hay que realizar un cálculo previo para conocer el momento cinético, que como se sabe está definido por la ecuación:

$$\mathbf{h} = \mathbf{r} \times \mathbf{v} \tag{3.26}$$

Conocido el momento cinético del movimiento orbital y a través de la ecuación de la excentricidad calculamos el vector excentricidad, para posteriormente obtener la propia excentricidad de la órbita gracias al módulo.

$$\mathbf{e} = \frac{\mathbf{v} \times \mathbf{h}}{\mu} - \frac{\mathbf{r}}{r} \quad \longrightarrow \quad e = |\mathbf{e}| \tag{3.27}$$

Si consideramos ahora un vector, **k**, normal al plano de referencia (véase la Figura 3.10), puede calcularse la inclinación a partir del producto escalar de dicho vector con el vector del momento cinético (normal al plano orbital).

$$\mathbf{h} \cdot \mathbf{k} = h.k. \cos i \quad \longrightarrow \quad i = \arccos \frac{h_z}{h} \tag{3.28}$$

Siendo h_z la componente normal del momento cinético respecto al plano de referencia. El resultado para la inclinación no requiere modificaciones adicionales puesto que debe estar contenido en el primer o segundo cuadrante.

Para tener situado el plano orbital respecto a una dirección de referencia (primer punto de Aries) utilizamos Ω (RAAN) cuyo cálculo es simple si obtenemos el vector **n** representado en la Figura 3.10. El proceso sería el siguiente:

$$\mathbf{n} = \frac{\mathbf{k} \times \mathbf{h}}{\|\mathbf{k} \times \mathbf{h}\|} = \begin{pmatrix} \cos\Omega \\ \sin\Omega \\ 0 \end{pmatrix} \quad \longrightarrow \quad \Omega = \arccos n_x \tag{3.29}$$

Puesto que Ω puede estar en el intervalo $[0, 2\pi]$ habrá que tener en cuenta también la componente n_y para conocer el cuadrante en el que está situado el resultado.

El siguiente elemento, ω, nos permite situar la órbita en el plano orbital y se va a obtener utilizando de nuevo un producto escalar entre los vectores **n** y **e**. Si se despeja:

$$\omega = \arccos \frac{\mathbf{e} \cdot \mathbf{n}}{e} \tag{3.30}$$

En este caso, en la elección del cuadrante, habrá que tener en cuenta la componente e_z del vector excentricidad de forma que si $e_z > 0$ entonces ω está en el intervalo $[0,\pi]$, mientras que si $e_z < 0$ estará en $[\pi,2\pi]$.

En último lugar, θ, el ángulo que permite posicionar el cuerpo en la órbita, va a calcularse mediante un proceso similar al anterior. Se despeja a partir del producto escalar entre **r** y **e**.

$$\theta = \arccos \frac{\mathbf{r} \cdot \mathbf{e}}{r.e} \tag{3.31}$$

Respecto al cuadrante en el que estará la anomalía verdadera calculada debe tenerse en cuenta el signo del producto $\mathbf{r} \cdot \mathbf{v}$, es decir, si es positivo se tendrá una anomalía verdadera en el intervalo $[0,\pi]$ y si es negativo en $[\pi,2\pi]$.

3.2.2 Conversión de elementos orbitales a posición y velocidad

Previamente a comentar el método de conversión se exponen brevemente los dos sistemas de referencia que van a utilizarse. Por un lado, el sistema de referencia perifocal es aquel que está centrado en el foco, con el eje OX^F apuntando hacia periapsis (en el plano orbital), el eje OZ^F paralelo al vector del momento cinético y el eje OY^F completando el triedro a derechas en el plano orbital (véase la Figura 3.11). Por otro lado, el sistema de referencia heliocéntrico (o geocéntrico ecuatorial) tiene su origen también en el foco, el Sol (o la Tierra), el eje OX^R apuntando hacia el primer punto de Aries en el plano eclíptico (o ecuatorial), el eje OZ^R paralelo al vector del momento cinético de los planetas (o coincidente con el eje de rotación de la Tierra) y el eje OY^R completando el triedro a derechas en el plano eclíptico (o en el plano ecuatorial) (véase la Figura 3.12).

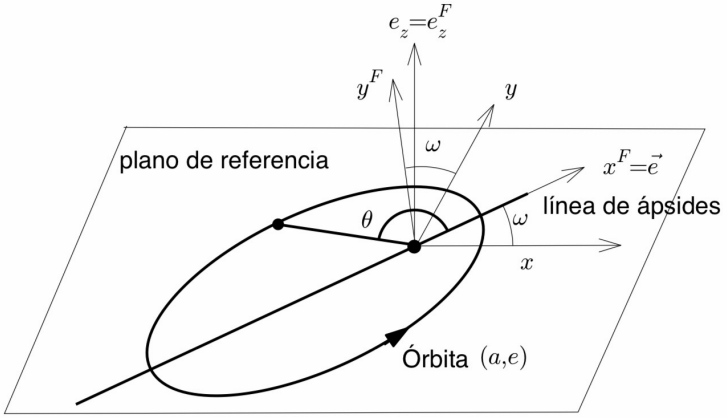


Figura 3.11 Representación del sistema de referencia perifocal. Extraído de [14].

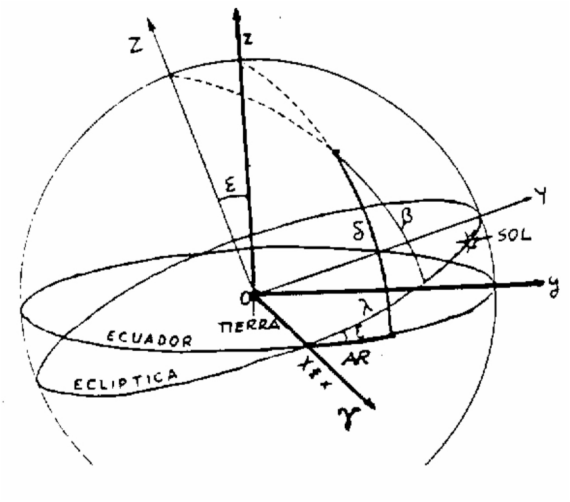


Figura 3.12 Representación del sistema de referencia geocéntrico ecuatorial. Extraído de [14].

Esta conversión requiere de un proceso más tedioso ya que hay que pasar los vectores posición y velocidad del sistema de referencia perifocal (F) al heliocéntrico (R) o al geocéntrico ecuatorial, según cual sea el foco de la órbita, mediante matrices de giros. En primer lugar se calculan los vectores de posición y velocidad a partir de los elementos orbitales en el sistema de referencia perifocal, para lo cual es necesario tener en cuenta las siguientes definiciones:

$$\mathbf{r} = r \cdot \mathbf{e}_r, \quad \mathbf{v} = \dot{r}\mathbf{e}_r^F + r\dot{\theta}\mathbf{e}_\theta^F, \quad \mathbf{e}_r^F = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix}, \quad \mathbf{e}_\theta^F = \begin{pmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{pmatrix} \tag{3.32}$$

Si a esto añadimos la ecuación general de una cónica,

$$r = \frac{p}{1 + e\cos\theta}, \tag{3.33}$$

y la relación entre el parámetro, p, de una cónica y el momento cinético,

$$p = \frac{h^2}{\mu}, \tag{3.34}$$

se está en condiciones de obtener la posición y velocidad en el sistema de referencia perifocal en función de los elementos orbitales:

$$\mathbf{r}^F = \frac{p}{1 + e\cos\theta} \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} \quad \text{y} \quad \mathbf{v}^F = \sqrt{\frac{\mu}{p}} \begin{bmatrix} -\sin\theta \\ e + \cos\theta \\ 0 \end{bmatrix} \tag{3.35}$$

Teniendo presente cómo se definen los sistemas de referencias, perifocal y heliocéntrico, puede deducirse que se requieren tres giros respecto a tres ejes diferentes para llegar de uno a otro. Por ejemplo, para pasar del heliocéntrico al perifocal (teniendo presente la Figura 3.10, se siguen los siguientes pasos:

1. Giro de un ángulo igual a Ω respecto al eje OZ^R
2. Giro de un ángulo igual a i respecto al eje X resultante del giro anterior
3. Giro de un ángulo igual a ω respecto al eje Z resultante del giro anterior

Cada giro resultará en una matriz de giro que multiplicará a los vectores obtenidos y permitirá cambiar su sistema de referencia. Para expresarlos en función del sistema heliocéntrico, que es lo que interesa, se sigue el proceso inverso al descrito. Llamando C_F^R a la matriz que incluye los tres giros para pasar del sistema perifocal a heliocéntrico, los vectores posición y velocidad del cuerpo que orbita serán:

$$\mathbf{r}^R = \frac{p}{1+e\cos\theta} C_F^R \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} \quad \text{y} \quad \mathbf{v}^R = \sqrt{\frac{\mu}{p}} C_F^R \begin{bmatrix} -\sin\theta \\ e+\cos\theta \\ 0 \end{bmatrix} \tag{3.36}$$

Si se quiere profundizar en el tema y comprobar cómo se calcula la matriz C_F^R puede verse los apuntes de Rafael Vázquez [14].

3.3 Maniobras orbitales

Debido a diversos factores como la imposibilidad de situar un satélite en su órbita justo tras su lanzamiento, el efecto de las perturbaciones que sufre un objeto que orbita o simplemente la necesidad de cambiar de órbita puesto que se quiere llevar a cabo una cierta misión provocan que la realización de maniobras orbitales sean esenciales en cualquier misión espacial que se desee desarrollar. En el modelo utilizado para este trabajo aparecen las maniobras orbitales en las etapas inicial y final de la misión puesto que permiten, en un primer instante, escapar de una órbita de aparcamiento entorno a la Tierra (típicamente usada) mediante un cierto impulso y a la llegada al cuerpo celeste final se ejecuta un segundo impulso de frenada que permite quedarse orbitando a su alrededor. Además, según los tiempos de vuelo que se introduzcan en el vector de decisión puede llevar a realizar algunas maniobras adicionales en el perigeo de la órbita hiperbólica durante la maniobra asistida por gravedad, aumentando por tanto el valor de la función objetivo. Puesto que en el Apartado 3.4 se trata el tema de las misiones interplanetarias se expondrán ahí el concepto de órbita de aparcamiento.

Las maniobras citadas van a ser maniobras básicas instantáneas y coplanarias con el resto de planetas que permitirán el paso de una órbita inicial a una final en la que se está interesado porque permite alcanzar otro planeta que nos ayude a ahorrar en consumo de combustible debido a la posibilidad de realizar maniobras asistidas por gravedad. Estas maniobras instantáneas pueden describirse sencillamente por un triángulo de velocidades, el cual muestra la relación entre la velocidad inicial, V_i, la velocidad final, V_f, y el impulso, .

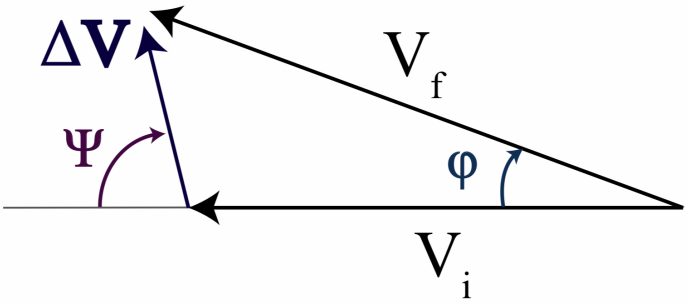


Figura 3.13 Triángulo de velocidades de una maniobra instantánea. Extraído de [14].

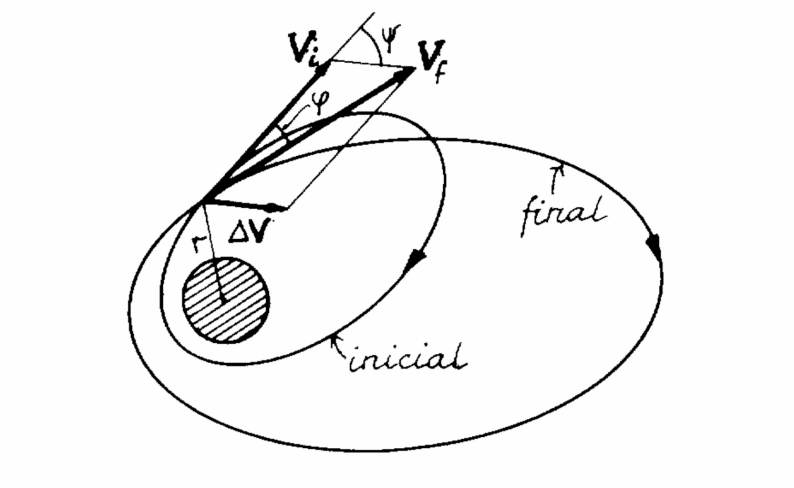


Figura 3.14 Esquema básico de maniobra instantánea. Extraído de [14].

Por trigonometría básica (teoremas del seno y del coseno) o haciendo uso de fasores puede resolverse el triángulo de velocidades mostrado en la Figura 3.13, generalmente en dos casos:

1. Se está en una cierta órbita y, tras un impulso dado $(\Delta V, \Psi)$, quiere calcularse la órbita final (V_f, ψ) mediante la conversión a elementos orbitales.
2. Se conocen las órbitas inicial y final y se pretende calcular el impulso necesario en algún punto de corte de las órbitas.

Este segundo punto es el que suele darse con más frecuencia en el diseño de misiones interplanetarias por el hecho de tener conocimiento previo de las órbitas de transferencias que se pretenden usar y las propias de los planetas. A través de dichas órbitas, se obtienen los valores de las velocidades en cada punto de esta por conversión como ya se ha expuesto anteriormente.
Finalmente añadir que, pese a que solo se han implementado en el modelo utilizado maniobras coplanarias de cambio de órbita, existen muchas otras, que podrían incluso mejorar los resultados obtenidos para este modelo si la misión lo requiriese, como:

- Rotación de línea de ápsides
- Cambio de inclinación de la órbita
- Cambio de la ascensión recta del nodo ascendente

El conjunto de estas maniobras orbitales permiten que la transferencia entre la órbita de partida y la de llegada sea como más convenga a la propia misión. Se tiene una gran diversidad de transferencias que se han ido optimizando con el paso del tiempo. Las más conocidas son: la transferencia de Hohmann, la bielíptica o biparabólica, transferencias con cambio de plano, *rendezvous*, etc.

3.3.1 Maniobras en espacio profundo

Este tipo de maniobra hace referencia a los impulsos que se realizan cuando la nave está viajando por el espacio camino a su destino en una órbita heliocéntrica. Las maniobras en espacio profundo pueden realizarse para llevar a cabo ciertas correcciones en la órbita de transferencia, un aumento de la velocidad que reduzca el tiempo de la misión, etc. Puesto que en este trabajo únicamente se analizan problemas bidimensionales, las maniobras quedan definidas por el triángulo de la Figura 3.13. Conocida la órbita inicial y la posición de la nave, puede obtenerse la velocidad con la que viaja. Tras realizar el impulso, resolviendo el triángulo de velocidades, se tendrá la nueva velocidad

con la posición que permanece igual (supuesto un impulso instantáneo). Con estas dos magnitudes podrá, mediante las conversiones expuestas, calcularse los elementos orbitales de la órbita que va a seguir la nave. Con esto, quedaría resuelta la maniobra en espacio profundo.

3.4 Misiones interplanetarias

Las misiones interplanetarias son el origen de los futuros viajes interplanetarios que realizarán personas altamente cualificadas (ingenieros, científicos, etc.) en no mucho tiempo puesto que ya están en desarrollo proyectos como el de la nave *Starship* que permitirá la llegada de una pequeña colonia de humanos a Marte por primera vez. Estas misiones han posibilitado la recogida de información suficiente de los diferentes cuerpos celeste (principalmente la Luna y Marte) que forman el Sistema Solar para tener un amplio conocimiento sobre las condiciones a las que se enfrentaría un humano en ese nuevo ambiente extraterrestre.

Las misiones realizadas hasta la fecha han tenido diversos objetivos: realizar un sobrevuelo a un planeta, mantenerse en una órbita entorno a un planeta, impactar contra algún cuerpo o, si se ha enviado un robot, llegar a la superficie en condiciones para realizar alguna otra misión de investigación. Todas permiten ampliar el conocimiento que tenemos del Sistema Solar. Para el caso en el que el objetivo principal de la misión no sea orbitar un cierto planeta, P, puede hacerse uso de las maniobras asistidas por gravedad que conllevan algunas ventajas como el poder obtener alguna información extra al sobrevolar P y recibir un impulso extra gratuito que permita ahorrar combustible. Sin embargo, como consecuencia de ello, se tardará algo más de tiempo en alcanzar el objetivo final de la misión aunque suele convenir hacer múltiples sobrevuelos previamente como se demuestra en los problemas estudiados en este trabajo.

En el ámbito de las misiones interplanetarias se tienen una serie de conceptos importantes que van a explicarse a continuación: la esfera de influencia, el ajuste de cónicas y la maniobra asistida por gravedad.

3.4.1 Esfera de influencia

La idea de la esfera de influencia parte de la necesidad de simplificar el comportamiento que tiene un cierto objeto (sonda, nave, cometa, etc.) que viaja por el espacio y que está influenciado por todas y cada una de las masas del Universo. Puesto que esto es imposible de modelar, por un lado se utiliza la hipótesis del problema de los tres cuerpos: Sol, planeta y sonda; sin tener en cuenta el resto de planetas del Sistema Solar y por otra parte se utilizan las esferas de influencia.

Las esferas de influencia determinan, por ejemplo para el caso del Sistema Solar, cuál es la región del espacio en la que el efecto de un cierto planeta sobre el objeto que viaja predomina frente al efecto del Sol. El radio de estas esferas de influencia de un planeta puede calcularse a través de la expresión:

$$R_{e_P} = L_P \left(\frac{\mu_p}{\mu_\odot} \right)^{2/5}, \tag{3.37}$$

donde L_P es la distancia que separa el centro del Sol del centro del planeta P y μ suele aproximarse como el producto de la constante de gravitación universal y la masa del cuerpo al que se refiera ($\mu_c = G \cdot m_c$).

Puesto que las distancias a cada planeta desde el Sol son enormemente mayores que los radios de las esferas de influencia de cada uno de ellos (véase la Figura 3.15 extraída de [14]) tiene sentido

la hipótesis de que a escala heliocéntrica las esferas de influencia quedan reducidas a un punto coincidente con la posición del planeta, no influyendo al objeto que viaja hasta que no alcanza dicha posición. Por otra parte, como el radio de dicha esfera es mucho mayor que el radio de cada planeta, en el caso planetocéntrico se entiende la esfera de influencia como una esfera de radio infinito de la cual se entra o sale por órbitas parabólicas o hiperbólicas. Luego, en el estudio de las misiones interplanetarias, se tienen diferentes etapas heliocéntricas y planetocéntricas que se van sucediendo la una a la otra para el caso de los sobrevuelos a varios planetas en una misma misión. Este concepto da lugar al desarrollo de un método que permita enlazar ambas fases: el ajuste de cónicas.

	radio esfera infl. ($km\times10^6$)	distancia media ($km\times10^6$)	radio planetario (km)
MERCURIO	0.11	58	2439
VENUS	0.62	108	6051
LA TIERRA	0.93	150	6378
MARTE	0.58	228	3397
JÚPITER	48	778	71398
SATURNO	55	1426	60330
URANO	52	2868	25400
NEPTUNO	87	4494	24300
PLUTÓN	36	5896	1600

Figura 3.15 Comparativa de las distancias al Sol, el radio de las esferas de influencia y los radios planetarios de cada planeta del Sistema Solar.

3.4.2 Ajuste de cónicas

Este concepto permite el cambio de sistema de referencia de la velocidad del cuerpo que viaja cuando entra o sale de la esfera de influencia de algún planeta. Vamos a tener algunos puntos en los que cambie el tipo de órbita debido a que, a partir de ellos, el foco que define la órbita descrita cambia y pasa a estar en la posición del planeta en lugar de en la del Sol para el caso en el que el cuerpo se acerca al planeta, pero también puede ocurrir lo contrario.

Si consideramos una misión sencilla que parta desde la Tierra (desde una órbita de aparcamiento) y llegue a cualquier otro planeta, va a ser necesario realizar un ajuste de cónicas para pasar de la hipérbola de escape de la esfera de influencia de la Tierra a la órbita elíptica heliocéntrica que recorrerá la nave hasta el planeta destino. Una vez alcanzado el planeta, visto a escala heliocéntrica, habrá que realizar un nuevo ajuste de cónicas para obtener ahora la velocidad correspondiente con la que se llega a la hipérbola con foco en el centro del nuevo planeta.

Puesto que, para los problemas que se pretenden optimizar, los tiempos de vuelo entre planetas son las variables de entrada en la optimización y los radios de las órbitas de los planetas son conocidos (se hace la hipótesis de órbitas circulares), siempre va a ser posible el cálculo de estas velocidades de partida y llegada en el sistema heliocéntrico gracias a que, resolviendo el problema de Lambert previamente, se habrá calculado la órbita correspondiente. Además, al ser conocidas las velocidades de los planetas, la aplicación del ajuste de cónicas no va a plantear ningún tipo de problema.

Si se sigue con el ejemplo de la misión sencilla comentada, los ajustes de cónicas en cada cambio de sistema de referencia quedarían de la siguiente forma:

- Salida de la esfera de influencia de la Tierra:

$$\mathbf{v}^{\odot}_{salida} = \mathbf{v}^{\odot}_{\oplus} + \mathbf{v}^{\oplus}_{\infty} \tag{3.38}$$

- Llegada a la esfera de influencia del planeta:

$$\mathbf{v}^{planeta}_{\infty} = \mathbf{v}^{\odot}_{llegada} - \mathbf{v}^{\odot}_{planeta} \tag{3.39}$$

Respecto a la notación utilizada, los superíndices aluden al sistema de referencia respecto al que se mide la velocidad y los subíndices definen cuál es la velocidad a la que se refiere. El subíndice ∞ indica que es la velocidad de salida o llegada a la hipérbola planetocéntrica. Por último, destacar que el radio de máximo acercamiento de la hipérbola de sobrevuelo de un planeta es un parámetro a determinar que se alcanza mediante la realización de unas pequeñas correcciones durante la etapa heliocéntrica y que no afectan al resultado final.

3.4.3 Maniobra asistida por gravedad

Esta maniobra, a diferencia de las ya explicadas, permite un cambio en la velocidad relativa respecto al Sol sin la necesidad de gastar combustible para ello. De ahí su gran utilidad para misiones que tengan que llegar a los planetas más alejados del Sistema Solar o, incluso, salir de este.

Para realización de una maniobra de este estilo basta con acercarse al planeta lo suficiente de manera que se entre en la esfera de influencia del mismo. Debido a las altas velocidades con las que se llegan a dicho punto de entrada en la esfera de influencia, la órbita recorrida en el nuevo sistema planetocéntrico será una hipérbola por lo que tras un cierto tiempo el objeto habrá pasado cerca del planeta y saldrá por la otra rama de la hipérbola con la misma velocidad en módulo (si no se realizan impulsos adicionales) que con la que entró. Sin embargo, la velocidad heliocéntrica resultante a la salida será totalmente diferente a la de la entrada porque también va a verse afectada por la velocidad del propio planeta (véase la Ecuación 3.38).

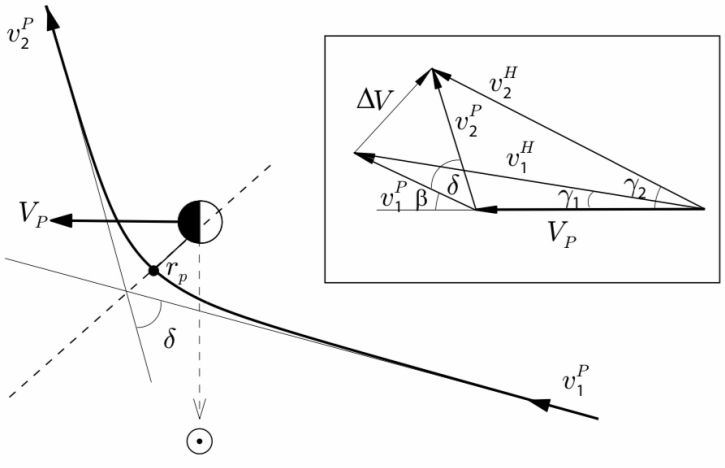


Figura 3.16 Esquema simple de una maniobra asistida por gravedad realizada por la cara iluminada. Extraído de [14].

Como se observa en la Figura 3.16, para el cálculo de la velocidad heliocéntrica a la salida de la esfera de influencia basta con resolver por trigonometría los triángulos mostrados, los cuales surgen del ajuste de cónicas. El incremento de velocidad que proporciona el sobrevuelo del planeta se

debe a que el propio planeta pierde parte de su energía cinética que ira a parar al objeto en órbita aumentando, por tanto, su velocidad. Dicho incremento de velocidad viene dado por la ecuación:

$$\Delta V = 2v_\infty \frac{1}{1+\frac{r_p v_\infty^2}{\mu_p}} = 2v_\infty \sin\frac{\delta}{2}, \tag{3.40}$$

siendo $v_\infty = v_1^p = v_2^p$ la velocidad de llegada y salida por las asíntotas de la hipérbola con foco en el planeta y r_p es el radio de periapsis de la órbita. Puede verse, como se dijo en el apartado anterior, que el radio de periapsis no está definido en ningún momento para este modelo, por lo que tendrá que ser una de las variables que se fijen antes de resolver el problema.

Es importante comentar que otra de las decisiones a tomar durante el planteamiento de una maniobra asistida por gravedad es la de elegir la configuración en la que se realiza la maniobra, es decir, si el sobrevuelo va a realizarse por la cara iluminada del planeta o por la cara oscura. En ambos casos el factor determinante en la elección será el conocer hacia donde se quiere que se desvíe nuestra nave y si se requiere frenar o acelerar, para ir hacia el interior o el exterior del Sistema Solar respectivamente. Para ello se resuelven ambas configuraciones y se escoge aquella que proporcione la velocidad heliocéntrica a la salida deseada.

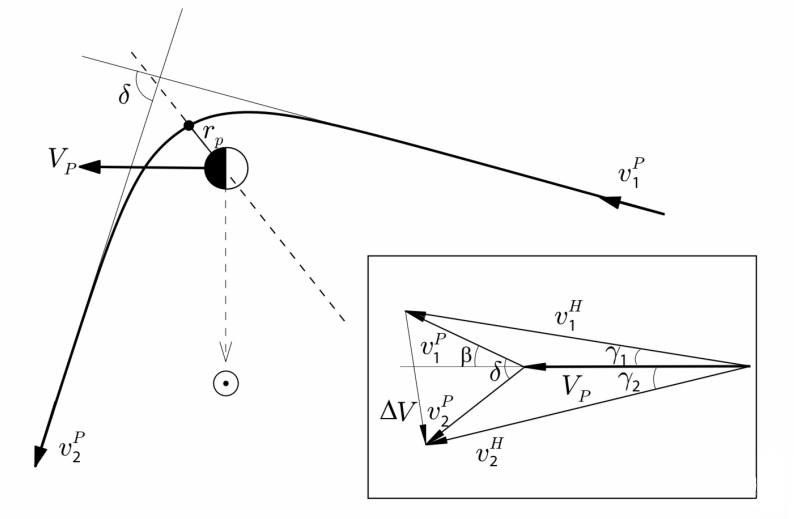


Figura 3.17 Esquema simple de una maniobra asistida por gravedad realizada por la cara oscura. Extraído de [14].

Finalmente, respecto al modelo utilizado en MATLAB para el cálculo de la maniobra asistida por gravedad decir que lo que calcula es si es necesario o no un impulso adicional durante esta maniobra y, en el caso de que lo sea, obtiene su magnitud. Esto se debe a que los datos de entrada en este caso van a ser las velocidades de salida y de entrada de la esfera de influencia puesto que se conocen los tiempos de vuelo entre cada planeta y, resolviendo el problema de Lambert asociado, las órbitas de transferencia necesarias. Para conseguir que la velocidad a la salida que se proporciona como dato se corresponda con la velocidad a la salida calculada a partir de la velocidad a la entrada, teniendo en cuenta un radio de periapsis fijado con anterioridad y resolviendo los triángulos de velocidades en la configuración más conveniente, es posible que aparezca la necesidad de realizar un pequeño impulso durante el sobrevuelo que añadirá el propio programa al total de impulsos realizado.

4 Problemas a resolver

Para el estudio y la puesta en funcionamiento de los algoritmos expuestos en el Capítulo 2 se han hecho uso de algunos problemas presentados por la ESA en 2008 a modo de competición para los investigadores que quisieran optimizar dichos problemas con algoritmos ya existentes o utilizar los suyos propios. Se han escogido algunos de estos problemas puesto que los modelos, a priori, complicados se dieron como dato para centrar la atención únicamente en la optimización.

Se conoce que cualquier problema de diseño de trayectoria debe pasar por un proceso exhaustivo de optimización para minimizar al máximo los recursos necesarios. Este tipo de problema de optimización está incluido en lo que se conoce como *Problemas de Optimización Global*, que se basan en encontrar un mínimo (o un máximo) absoluto en un cierto espacio de búsqueda restringido por una serie de límites para una función no lineal. Parece obvio pero a su vez es de gran importancia el usar siempre el mismo modelo para un cierto problema si se quieren comparar diferentes métodos de optimizar la solución puesto que será fuertemente dependiente de cualquier detalle del modelo, de las restricciones usadas, etc.

4.1 Los modelos

A modo de introducción puede comentarse que las trayectorias a optimizar van a incluir múltiples maniobras asistidas por gravedad y se presentará algún problema que también implemente maniobras en espacio profundo para comparar los tiempos requeridos al pretender encontrar soluciones cercanas a las óptimas presentadas en la página de la ESA [4]. Los problemas que no incluyen maniobras en espacio profundo representan una cierta misión donde el vehículo espacial solo tiene la capacidad de generar un incremento en su velocidad durante la fase planetocéntrica en el paso cercano a algún cuerpo masivo del Sistema Solar. Como se verá posteriormente, este tipo de problemas son de menor dimensión que los que incluyen maniobras en espacio profundo debido a que en los segundos se debe optimizar además el punto en el que debe producirse dicho impulso.

De forma general, cualquier modelo a optimizar seguirá el siguiente patrón:

$$\begin{aligned} &\text{Encuentra: } x \in \mathbb{R}^n \\ &\text{Minimiza: } J(x) \\ &\text{Sujeto a: } g(x) \end{aligned} \tag{4.1}$$

donde x es el vector decisión, es decir, es el vector que incluye las variables que van a calcularse durante la optimización; J es la función objetivo, la cual devuelve un cierto valor dado un vector de decisión; y g son el conjunto de restricciones no lineales que dependerán del problema a resolver.

En cuanto a los vectores de decisión habrá un clara diferencia según el tipo de problema que se esté tratando. Si se considera que la misión pasa por una secuencia de N planetas, los vectores de decisión tendrán las siguientes estructuras:

- **Sin maniobra en espacio profundo**:

$$x = [t_0, T_1, ..., T_{N-1}] \tag{4.2}$$

- **Con maniobra en espacio profundo**:

$$x = [t_0, V_\infty, u, v, \eta_1, T_1, r_{p2}, b_{incl2}, \eta_2, T_2, ..., r_{p_{N-1}}, b_{incl_{N-1}}, \eta_{N-1}, T_{N-1}] \tag{4.3}$$

Puede observarse claramente la diferencia de longitud entre los dos vectores de decisión para una misma secuencia de planetas. Respecto a las variables tenemos las siguientes:

- t_0: Fecha de comienzo de la misión
- V_∞, u y v: Definen la velocidad heliocéntrica a la salida de la hipérbola planetocéntrica
- T_N y η_N: Definen, respectivamente, el tiempo que tarda el vehículo en alcanzar el planeta N partiendo del $N-1$ y en qué fracción del viaje se realiza una maniobra de espacio profundo.
- r_{p_N}: Radio mínimo de acercamiento al planeta N
- b_{incl_N}: Ángulo de rotación de la velocidad de escape de una órbita hiperbólica planetocéntrica respecto a la velocidad de entrada en ella. (Ángulo γ en los desarrollos)

Por último, comentar que la función objetivo típica es la minimización del combustible consumido durante la misión, aunque puede haber otras como la minimización del tiempo total de vuelo, la maximización de la velocidad de impacto contra un asteroide, etc. Es por ello que las restricciones no lineales dependen de cada problema considerado.

4.2 Problemas

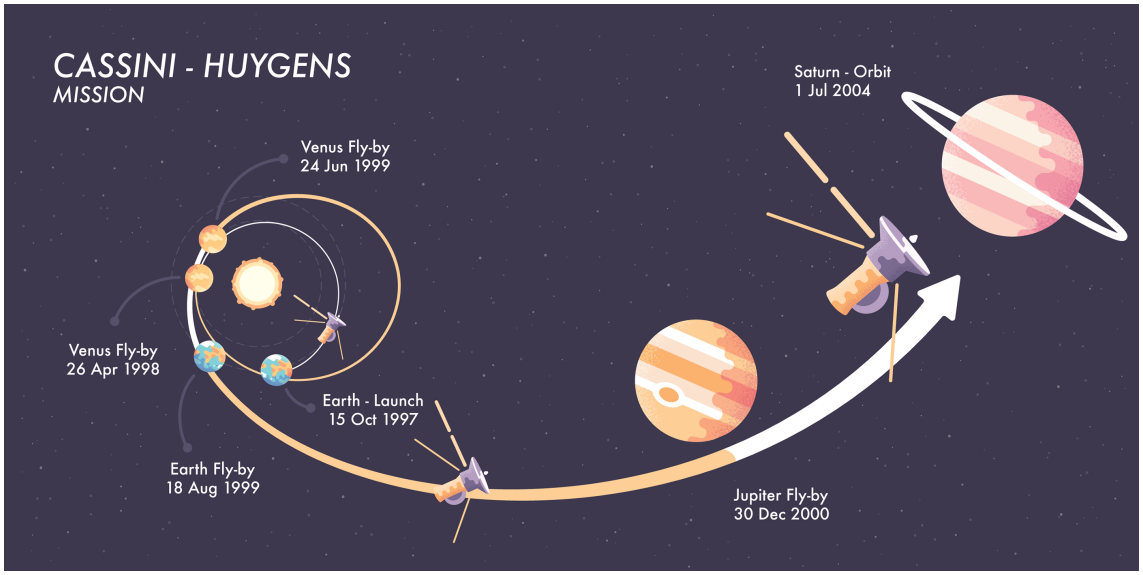
Los códigos de modelización de las trayectorias interplanetarias de las misiones que van a optimizarse, es decir, los que implementan todo lo comentado en el Capítulo 3 están publicados en la página de la ESA [4] y son los archivos de MATLAB: *mga.m* para los problemas sin maniobras en espacio profundo y *mga_dsm.m* para los problemas que incluyen maniobras en espacio profundo.

4.2.1 Cassini1

Este problema hace referencia a la clásica misión Cassini/Huygens, un proyecto de la NASA, la ESA y la ASI cuyo objetivo era estudiar el planeta Saturno y sus satélites naturales (lunas). La misión no tripulada tuvo su lanzamiento en octubre del 1997 desde Cabo Cañaveral y entró en la órbita de Saturno en julio del 2004. Tras un tiempo en órbita, la sonda se separó de la nave para alcanzar a su luna Titán y cuando lo consiguió el módulo de descenso (Huygens) inició su operación de bajada a la superficie para recoger información. Antes de su desintegración en la atmósfera de Saturno por el agotamiento de todo el combustible y la pérdida del control a finales de 2017, en abril de ese mismo año la sonda consiguió orbitar entre los anillos y el planeta recogiendo sus últimos datos.

Pese a que la misión completa duró 20 años, en el problema que se pretende optimizar solamente se modela el viaje desde la Tierra hasta Saturno teniendo únicamente en cuenta las maniobras asistidas por gravedad que realizó durante el trayecto, es decir, sin realizar maniobras en espacio profundo. La secuencia de planetas que visitó es la siguiente:

1. Tierra (salida)
2. Venus
3. Venus
4. Tierra
5. Júpiter
6. Saturno (llegada)



Autor: José Utreras

Figura 4.1 Ilustración del viaje que realizó la nave que transportó a la sonda hasta Saturno. Extraída de [13].

El objetivo de la misión a optimizar es la minimización de la suma de todos los impulsos (ΔV) realizados: el impulso de salida, los impulsos requeridos durante los sobrevuelos para alcanzar los planetas siguientes y el impulso final para mantenerse en la órbita deseada. Esta órbita final está caracterizada por un radio de periapsis (r_p) de 108950 km y una excentricidad (e) de 0.98.

Como se introdujo en el apartado de maniobra asistida por gravedad de la sección 3.4, los radios de máximo acercamiento al planeta durante las maniobras asistidas por gravedad deben ser datos y considerarse que durante el vuelo interplanetario se realizan pequeñas correcciones despreciables para cumplir dicha restricción. Los radios que se utilizan para este problema son los siguientes:

- Sobrevuelo a Venus: $r_p^{\varnothing} = 6351.8\ km$
- Sobrevuelo a la Tierra: $r_p^{\oplus} = 6778.1\ km$
- Sobrevuelo a Júpiter: $r_p^{\jupiter} = 671492\ km$

Por último, se presenta en la Tabla 4.1 el vector decisión (x), con el que se pretende minimizar la función objetivo (la suma de impulsos realizados), los límites inferiores y superiores de las distintas variables y las unidades en las que se miden.

Tabla 4.1 Variables de optimización para el problema de Cassini1.

Componente del vector	Variable	Lím. inferior	Lím. superior	Unidad
x(1)	t_0	-1000	0	MJD2000
x(2)	T_1	30	400	días
x(3)	T_2	100	470	días
x(4)	T_3	30	400	días
x(5)	T_4	400	2000	días
x(6)	T_5	1000	6000	días

Las variables que permitirán optimizar el problema son: t_0, el instante de partida en la época MJD2000, y T_1,..., T_5, que son los tiempos de vuelo en cada trayectoria heliocéntrica que realiza la nave en su viaje hasta Saturno, por ejemplo, T_1 será el tiempo de vuelo desde la salida de la Tierra hasta la llegada a Venus por primera vez.

4.2.2 GTOC1

En este caso, el problema que va a resolverse no hace referencia a ninguna misión real que se haya realizado, si no que está inspirado en la primera competición de optimización global de trayectoria (*Global Trajectory Optimisation Competition*) que se organizó. De nuevo, es un problema con múltiples maniobras asistidas por gravedad, sin maniobras en espacio profundo, en el que la secuencia de planetas que se sobrevuelan viene dada y es la que sigue:

1. Tierra (salida)
2. Venus
3. Tierra
4. Venus
5. Tierra
6. Júpiter
7. Saturno
8. Asteroide TW229 (llegada)

A diferencia del anterior, este problema trata de encontrar un máximo, más concretamente, pretende maximizar el cambio del semieje mayor de la órbita del asteroide TW229 a través de un choque de la nave contra el mismo. Para ello, se utiliza la función objetivo:

$$J(x) = m_f \mathbf{U} \cdot \mathbf{v},$$

la cual permite cuantificar cómo de energético va a ser el choque entre ambos cuerpos, siendo m_f la masa final de la nave espacial y las velocidades, $\mathbf{U}$ y $\mathbf{v}$, son, respectivamente, la del asteroide y la relativa del asteroide respecto de la nave.

Los datos característicos de este problema son el incremento de velocidad inicial que es necesario para escapar de la atracción terrestre, $\Delta V = 2.5\ km/s$, el impulso específico del motor cohete

instalado, $I_{sp} = 2500\ s$, y la masa inicial de la nave espacial, $m_0 = 1500\ kg$. Como ya se sabe, por estar presentes las maniobras asistidas por gravedad en este problema, también habrá que predefinir los radios de periapsis de las órbitas hiperbólicas que se van a describir dentro de las esferas de influencia de los planetas que se visitan. Estas distancias serán:

- Sobrevuelo a Venus: $r_p^{\venus} = 6351.8\ km$
- Sobrevuelo a la Tierra: $r_p^{\earth} = 6778.1\ km$
- Sobrevuelo a Júpiter: $r_p^{\jupiter} = 600000\ km$
- Sobrevuelo a Saturno: $r_p^{\saturn} = 70000\ km$

Finalmente, se presentan en la Tabla 4.2 las variables que forman ahora el vector de decisión y que permitirán la optimización del problema, junto con las restricciones y las unidades de medida.

Tabla 4.2 Variables de optimización para el problema de GTOC1.

Componente el vector	Variable	Lím. inferior	Lím. superior	Unidad
x(1)	t_0	3000	10000	MJD2000
x(2)	T_1	14	2000	días
x(3)	T_2	14	2000	días
x(4)	T_3	14	2000	días
x(5)	T_4	14	2000	días
x(6)	T_5	100	9000	días
x(7)	T_6	366	9000	días
x(8)	T_7	300	9000	días

4.2.3 Messenger

El problema que va a optimizarse es una simplificación de la misión Messenger, una sonda espacial no tripulada de la NASA que se lanzó en agosto de 2004 y que alcanzó una órbita entorno a Mercurio en marzo de 2011. Su objetivo principal era la observación orbital del planeta durante un año terrestre. La realidad es que su vida se alargó hasta cuatro años porque seguían obteniéndose datos nuevos y de interés para conocer mejor el planeta. Finalmente, acabado el proyecto, la sonda colisionó contra Mercurio en abril de 2015.

Como el resto de misiones planetarias realizadas, la misión Messenger también realizó diversos sobrevuelos a otros planetas antes de alcanzar su destino final. La secuencia de planetas seguida fue:

1. Tierra (salida)
2. Tierra
3. Venus
4. Venus
5. Mercurio (llegada)

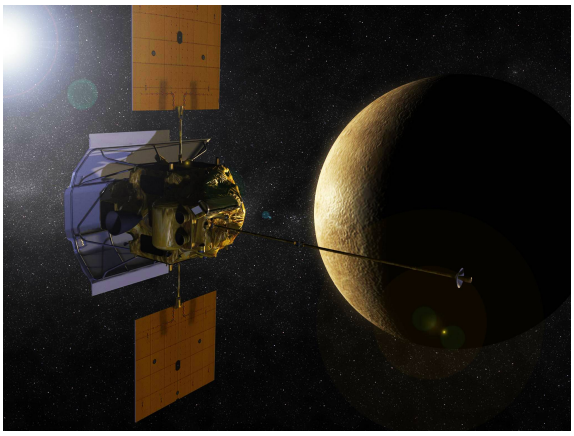


Figura 4.2 Representación ficticia de la sonda Messenger orbitando Mercurio.

El objetivo del problema a optimizar vuelve a ser la minimización de la suma de todos los impulsos realizados. En este caso, al impulso de salida y llegada se añade algún impulso extra debido a las maniobras en espacio profundo que se realizan. La órbita objetivo se catateriza por un rp y una e... Análogo al resto de problemas, los radios de periapsis de las órbitas hiperbólicas seguidas durante el sobrevuelo a los planetas deben ser dato. Nótese que, a continuación, se muestran los radios mínimos posibles, pero que en la Tabla 4.3 se exponen algunas de las variables del vector de decisión que definen realmente cuáles van a ser los radios de periapsis. Los mínimos serán:

- Sobrevuelo a Venus: $r_p^{\varphi} = 6351.8\ km$
- Sobrevuelo a la Tierra: $r_p^{\oplus} = 6778.1\ km$

Finalmente, en la Tabla 4.3 se muestran el vector decisión que permitirá minimizar la función objetivo, las restricciones para cada una de las variables y las unidades de las mismas.

Tabla 4.3 Variables de optimización para el problema Messenger.

Componente del vector	Variable	Lím. inferior	Lím. superior	Unidad
x(1)	t_0	1000	4000	MJD2000
x(2)	V_∞	1	5	km/s
x(3)	u	0	1	n/a
x(4)	v	0	1	n/a
x(5)	T_1	200	400	días
x(6)	T_2	30	400	días
x(7)	T_3	30	400	días
x(8)	T_4	30	400	días
x(9)	η_1	0.01	0.99	n/a
x(10)	η_2	0.01	0.99	n/a
x(11)	η_3	0.01	0.99	n/a
x(12)	η_4	0.01	0.99	n/a
x(13)	$\bar{r}_{p1}$	1.1	6	n/a
x(14)	$\bar{r}_{p2}$	1.1	6	n/a
x(15)	$\bar{r}_{p3}$	1.1	6	n/a
x(16)	b_{incl1}	-π	π	n/a
x(17)	b_{incl2}	-π	π	n/a
x(18)	b_{incl3}	-π	π	n/a

En cuanto a las variables vuelven a aparecer algunas de los problemas anteriores, t_0 y T_p, que son la época de partida de la Tierra y los tiempos de vuelo entre cada par de planetas. Junto a estas aparece V_∞, el módulo de la velocidad de escape de la órbita hiperbólica del primer planeta, con u y v, que proporcionan la dirección a la velocidad de salida de la esfera de influencia de la Tierra respecto a la velocidad de tralación de esta. Al tener maniobras en espacio profundo, surgen las variables $\eta_1, ..., \eta_4$, las cuales definen en que fracción del tiempo de vuelo entre un planeta y otro ocurre la maniobra. Ya se ha comentado que los sobrevuelos no van a realizarse con los radios mínimos de periapsis expuestos anteriormente, si no que se van a multiplicar por un cierto factor adimensional que los aumenta, $\bar{r}_{p1}, ..., \bar{r}_{p3}$. En último, lugar las variables $b_{incl1}, ..., b_{incl3}$, que hacen referencia al ángulo δ que rota la velocidad de salida de la órbita hiperbólica de los sobrevuelos respecto a la velocidad de entrada en la misma.

5 Estudio de los parámetros de cada algoritmo

En este capítulo se va a exponer el análisis realizado sobre cada uno de los parámetros que afectan a la actuación de cada algoritmo. Se verá, que debido a la aleatoriedad de estos, se ha llegado a un valor de cada parámetro característico tras la realización de 10 ejecuciones del algoritmo con condiciones idénticas para poder obtener una cierta tendencia de resultados.

Esta selección de parámetros se ha hecho tanto para los problemas sin maniobras como para el problema con maniobras en espacio profundo utilizando como problema de análisis el de *Cassini1*. El objetivo de este estudio era encontrar la configuración ideal de cada uno de los algoritmos para llegar a la mejor solución posible con tiempos de ejecución relativamente pequeños. En lo que sigue, se diferencia por secciones los estudios para los diferentes algoritmos.

5.1 Algoritmo genético

Como ya se sabe los parámetros que van a definir cómo va a ser la actuación del algoritmo genético son: el factor de mutación, el parámetro de cruce, el tamaño de la población y la población élite que evoluciona intacta con el paso de las generaciones. Para cada uno de ellos se ha realizado un estudio extenso en tiempo para obtener su valor más adecuado y que se utilizará para la optimización de los problemas escogidos.

5.1.1 Factor de mutación

Al comenzar el estudio por este parámetro, se tiene el problema de no conocer los valores óptimos del resto que se necesitan dar al algoritmo genético. Por ello y porque es necesario que estén definidos, se han elegido según unas ejecuciones preliminares hechas con el objetivo de ver cómo actúa de forma general el algoritmo genético cuando se varían bruscamente sus parámetros. Una vez probados algunos valores se eligen los siguientes:

- Parámetro de cruce: 0.8
- Tamaño de población: 400
- Población de élite: 5 % de toda la población

En cuanto a la condición de parada, en este caso, se ha escogido la limitación de generaciones máximas fijándola en 5000 generaciones para asegurar que el algoritmo consigue soluciones relativamente buenas pese a no tener los parámetros adecuadamente escogidos de momento.

El proceso seguido para el análisis se basa en la obtención del mejor valor encontrado por el algoritmo durante 10 ejecuciones para cada valor elegido del factor de mutación, junto con la media de la última generación, que no es más que la media de los valores de la función objetivo correspondientes a los vectores de decisión que conforman esa última generación. Este valor medio nos proporcionará información acerca de cómo está realizando la búsqueda el algoritmo, es decir, si tiende más a la explotación de una zona o a la exploración general del espacio de búsqueda.

Los valores escogidos para el factor de mutación, es decir, para la probabilidad de mutación se debe a que tiene que haber un equilibrio entre la mutación de los individuos buenos y malos. Si dicho factor es muy grande habrá mucha mutación en los individuos, especialmente en los buenos, y el algoritmo no converge, mientras que si el factor es muy pequeño hay poca mutación en los individuos, especialmente en los malos, el algoritmo tampoco evoluciona adecuadamente.

Tras un tiempo de 6.2523 horas, se han obtenido todos los valores presentados en la Tabla 5.1. Puede verse como la última fila hace referencia a la media de las 10 ejecuciones, la cual nos permitirá escoger el valor del factor de mutación más adecuado.

Tabla 5.1 Resultados obtenidos del estudio del factor de mutación para el algoritmo genético con el problema *Cassini1*.

MUTACIÓN	0,005	0.01	0.02	0.03
Ejecución 1	5.6983 (5.6984)	5.61354 (5.8462)	10.2818 (12.3222)	17.3491 (18.9825)
Ejecución 2	11.2056 (11.2057)	5.9002 (6.4487)	5.6039 (5.6641)	12.8217 (13.5637)
Ejecución 3	5.6486 (5.6486)	5.8334 (5.8335)	5.0925 (6.3104)	11.1557 (11.1608)
Ejecución 4	6.8339 (6.8587)	5.1569 (5.2737)	6.2239 (6.2243)	5.9588 (5.9711)
Ejecución 5	6.1623 (6.1624)	5.3099 (5.6429)	5.6823 (6.0184)	6.9743 (9.6357)
Ejecución 6	5.2824 (5.3590)	5.3982 (5.7039)	5.9163 (6.7651)	11.1726 (11.1727)
Ejecución 7	11.0356 (11.0357)	11.2784 (11.2862)	5.9990 (6.0223)	11.0181 (11.7492)
Ejecución 8	11.1247 (11.5017)	5.9603 (6.4661)	5.3360 (5.3691)	5.9967 (6.4028)
Ejecución 9	17.4356 (17.4359)	6.1677 (6.6327)	5.0941 (5.2225)	5.8357 (7.1539)
Ejecución 10	6.1344 (6.1345)	6.0915 (6.2419)	5.4794 (5.5028)	24.3946 (25.4705)
Media	8.6561 (8.7041)	6.2710 (6.4877)	6.0709 (6.5421)	11.2677 (12.1263)

Antes de comentar los resultados, aclarar que la presentación por la que se ha optado ha sido: *Mejor valor de la función objetivo (Media de la función objetivo en la última generación)*.

Una vez completada la tabla y representados los puntos en la Figura 5.1, pueden compararse los resultados para cada valor del factor de mutación, observándose como a medida que crece dicho factor la diferencia entre la media de la última generación y el mejor valor encontrado aumenta. Esto genera dos situaciones extremas diferenciadas por un valor alto y bajo del factor de mutación.

En el primer caso, puede verse que existe una cierta aleatoriedad de los mejores valores pero que la media es muy parecida a estos. Esto se debe a que para un factor de mutación bajo el algoritmo suele quedar atrapado en mínimos locales que, a veces, distan mucho de la solución óptima y no logra escapar porque la mutación no es suficiente.

En el segundo caso ocurre lo contrario, la media de la última generación es muy diferente al mejor valor, luego la mutación es excesiva y el algoritmo no tiene la capacidad de explotar una zona por lo que no consigue llegar ni siquiera a un mínimo local, si no que va obteniendo puntos con una cierta aleatoriedad por dicha mutación.

Un resultado importante a destacar es que con la variación del factor de mutación puede conseguirse que el algoritmo sea más aleatorio (factor alto) o menos (factor bajo). Esto puede ser una gran ventaja a la hora de aplicar varias veces el algoritmo genético ya que podría aplicarse inicialmente en su configuración más aleatoria y una vez obtenidos buenos resultados de la función objetivo aplicarse con un factor de mutación bajo y así explotar unas zonas concretas.

Para este trabajo, se ha decidido tomar un factor de los intermedios, concretamente, constante e igual a 0.02 puesto que es el caso para el que la media del mejor valor de la función objetivo es menor. Esto se debe a que el algoritmo combina adecuadamente las funciones de exploración y explotación por tener un factor de mutación equilibrado. Por último, en la Figura 5.2 están representados únicamente los mejores valores medios tras las 10 ejecuciones, lo cual confirma que la elección de un factor de mutación de 0.02 es la idónea.

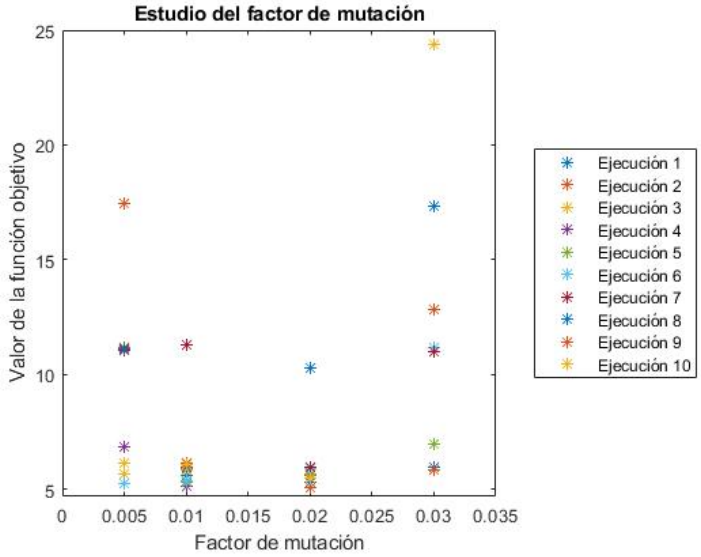


Figura 5.1 Representación gráfica de los valores obtenidos para las 10 ejecuciones.

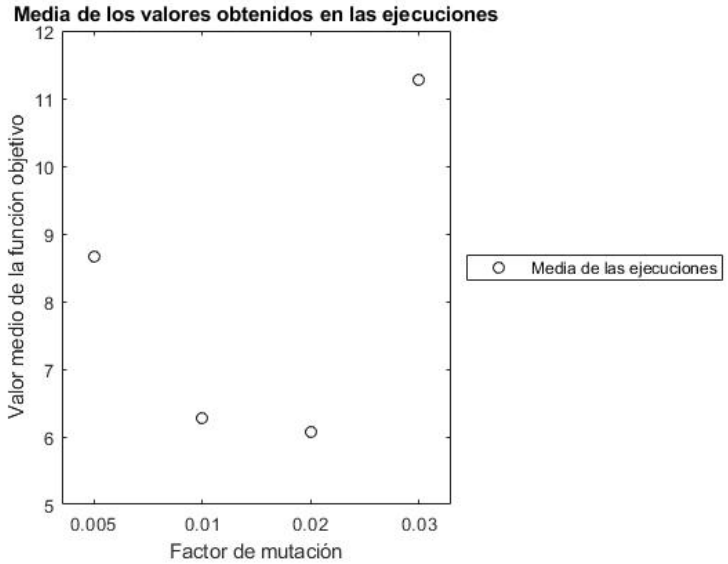


Figura 5.2 Representación gráfica de los valores medios obtenidos para las 10 ejecuciones.

Finalmente, para comprobar el nivel de aleatoriedad que tiene el algoritmo genético para un factor de mutación de 0.03 se ha programado un algoritmo de obtención de puntos aleatorios (véase la Sección B.4) y se ha ejecutado también 10 veces con el mismo número de evaluaciones para comparar los resultados. Viendo los datos generados de la Tabla 5.2 y los resultados para 0.03, se deduce como, efectivamente, dicho caso tiene un cierto comportamiento aleatorio y podría utilizarse para una exploración abierta del espacio de búsqueda.

Tabla 5.2 Resultados de las 10 ejecuciones del algoritmo aleatorio.

EJECUCIONES DE ALGORITMO ALEATORIO	1	2	3	4	5
Resultados	16.8045	8.2025	12.4322	8.3757	14.9847
	6	**7**	**8**	**9**	**10**
	7.0757	13.4286	7.2624	11.1007	10.8220

5.1.2 Parámetro de cruce

De nuevo, se va a comenzar exponiendo los valores utilizados de los diferentes parámetros del algoritmo para la obtención de los resultados que se presentan a continuación. En este caso, ya se tiene fijado uno de los parámetros, el factor de mutación, luego parece lógico utilizar el valor escogido. En cuanto al resto, se mantendrán los mismos utilizados en el estudio de la mutación. Igualmente cada ejecución tendrá un límite de 5000 generaciones, ya que son suficientes para ver la tendencia que siguen los resultados según cambie el parámetro de cruce. Por tanto, los parámetros usados son:

- Factor de mutación: 0.02
- Tamaño de población: 400
- Población de élite: 5 % de toda la población

Siguiendo un proceso de análisis similar de 10 ejecuciones (para cada valor a estudiar del parámetro de cruce) y el cálculo de la media de ellas será posible sacar algunas conclusiones que nos permitan elegir un valor. Tras un tiempo de 7.8054 horas, se han conseguido los mejores valores presentados en la Tabla 5.3, donde la última fila vuelve a ser la media para cada parámetro de cruce.

Tabla 5.3 Resultados obtenidos del estudio del parámetro de cruce para el algoritmo genético con el problema *Cassini1*.

CRUCE	0	0.25	0.5	0.75	1
Ejecución 1	11.7332	5.5689	5.7034	7.5293	7.0011
Ejecución 2	11.2187	5.3568	8.0663	5.7051	7.3322
Ejecución 3	11.4278	6.2473	12.7121	5.3703	14.2559
Ejecución 4	6.0939	11.5691	6.5199	11.5074	7.9990
Ejecución 5	15.6779	5.4226	5.9645	7.8283	8.2897
Ejecución 6	10.0443	8.5947	6.3057	5.3525	8.6287
Ejecución 7	12.3329	6.0333	5.3145	4.9897	12.3793
Ejecución 8	18.7022	6.4628	11.2675	5.4101	5.8105
Ejecución 9	5.7695	11.7016	5.6267	5.3240	12.0668
Ejecución 10	7.0367	11.1182	11.1580	5.4864	11.8515
Media	11.0037	7.8075	7.8638	6.4503	9.5615

Con los resultados de la Tabla 5.3, pueden distinguirse dos situaciones claras, aquellas en las que el parámetro vale 0 o 1 y en las que toma un valor intermedio a los anteriores.

Puede verse como para la primera situación, la media es propia de un algoritmo aleatorio, lo cual está provocado porque realmente se tienen unas condiciones de no cruce entre los progenitores para generar la descendencia. Puesto que la partición del cromosoma se hace antes del primer gen (valor 0) o después del último (valor 1) los descendientes son cromosomas idénticos a los progenitores desapareciendo el efecto de la evolución genética con las generaciones. Por ello, el resultado de la ejecución del algoritmo será el mejor resultado encontrado con la creación aleatoria de la primera generación, salvo que por alguna mutación se encuentre algún cromosoma mejor en alguna de las generaciones. Por ejemplo, para el caso del parámetro nulo se tendría lo siguiente:

- Progenitores:
$$[|aabccbcacb] \quad \text{y} \quad [bbccabacab|]$$
- Descendencia:
$$[bbccabacab|] \quad \text{y} \quad [|aabccbcacb]$$

En el caso de tener un parámetro de cruce del intervalo $(0,1)$, sí que habrá cruce de los progenitores generando una descendencia de cromosomas diferentes que permitirán la evolución del algoritmo hacia cromosomas cada vez mejores, o lo que es lo mismo, hacia soluciones más cercanas al óptimo del problema. En esta segunda situación la generación de descendencia se parece más a lo siguiente:

- Progenitores:
$$[aabccb|cacb] \quad \text{y} \quad [bbccab|acab]$$
- Descendencia:
$$[aabccb|acab] \quad \text{y} \quad [bbccab|cacb]$$

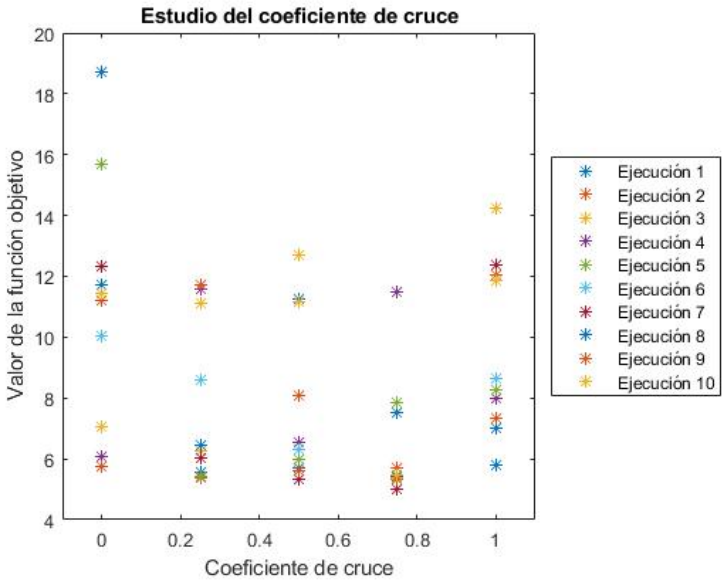


Figura 5.3 Representación gráfica de los valores obtenidos para las 10 ejecuciones.

Tras este análisis, respaldado por la Figura 5.3, está claro que se debe escoger un parámetro de cruce del intervalo $(0,1)$ puesto que de otro modo será prácticamente imposible encontrar el óptimo del problema o incluso un valor cercano.

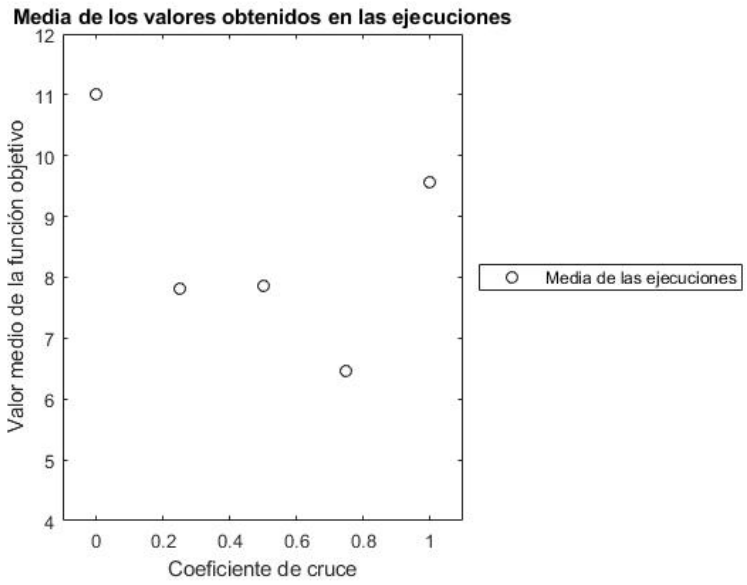


Figura 5.4 Representación gráfica de los valores medios obtenidos para las 10 ejecuciones.

Por tanto, viendo la Tabla 5.3 o la Figura 5.4, puede concluirse que el valor óptimo para la optimización del problema *Cassini1* será el de 0.75.

5.1.3 Tamaño de población

En el estudio del tamaño de la población que va a utilizarse durante las optimizaciones hay que encontrar un cierto equilibrio entre la obtención de buenos resultados y tiempos de ejecución razonables. Como se verá en los resultados cuanto mayor sea la población mejores valores de la función objetivo se conseguirán, pero con un costo temporal que aumenta linealmente con el tamaño (véase la Figura 5.5). Antes de todo, se van a presentar qué parámetros fijos se han utilizado para este análisis. Puesto que el factor de mutación y el parámetro de cruce ya han sido analizados, se les dará el valor que mejores resultados proporciona, mientras que la población de élite se mantendrá como en los análisis anteriores a falta de su estudio propio, por lo que se tendrá:

- Factor de mutación: 0.02
- Parámetro de cruce: 0.75
- Población de élite: 5 % de toda la población

Sin embargo, en este caso se va a cambiar el número de generaciones máximas reduciéndolas a 1000 generaciones, ya que se pretende comprobar las diferencias existentes, según el número de individuos, en la capacidad de evolucionar hacia resultados cercanos al óptimo en las primeras generaciones.

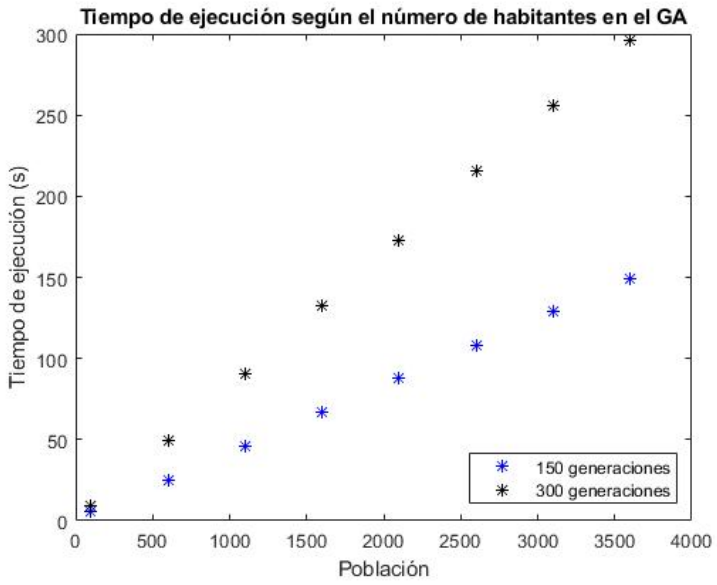


Figura 5.5 Representación gráfica del tiempo de ejecución según aumenta el tamaño de población.

Como ya se ha expuesto, en este caso también se seguirá un análisis de la tendencia de los resultados durante 10 ejecuciones para cada tamaño de población a considerar. Los tamaños se han ido incrementando hasta el punto en el que no se mejoraba los resultados obtenidos con un tamaño menor, ya que lo único que proporcionaban era un aumento del tiempo de ejecución. Tras 3.1512 horas de trabajo, se completa la Tabla 5.4 que permitirá la elección de un tamaño adecuado.

La presentación de los datos de la Tabla 5.4 se realiza según la estructura: *Mejor valor de la función objetivo (Tiempo de ejecución de las 1000 generaciones)*.

Tabla 5.4 Resultados obtenidos del estudio del tamaño de población para el algoritmo genético con el problema *Cassini1*.

TAMAÑO	100	250	500	750	1000	1500
Ejecución 1	5.8477 (0.4733)	11.0386 (1.1584)	17.5351 (2.3211)	7.7826 (3.4572)	5.0514 (4.6378)	5.6287 (6.9235)
Ejecución 2	19.0240 (0.4667)	11.3273 (1.1537)	18.9817 (2.3171)	5.0810 (3.4435)	5.3539 (4.6090)	5.7012 (6.8814)
Ejecución 3	11.4837 (0.4672)	21.7587 (1.1600)	5.5010 (2.2986)	15.1879 (3.4603)	11.1488 (4.6163)	13.7126 (6.8765)
Ejecución 4	19.7266 (0.4641)	15.6114 (1.1588)	11.4711 (2.3117)	5.3133 (3.4423)	5.2171 (4.5912)	5.3066 (6.9514)
Ejecución 5	17.9652 (0.4671)	5.7691 (1.1445)	10.9979 (2.3179)	5.7519 (3.4413)	6.0061 (4.6003)	5.4582 (6.9003)
Ejecución 6	18.6449 (0.4698)	13.3080 (1.1401)	7.2684 (2.3185)	5.6912 (3.4345)	5.4924 (4.6049)	5.0848 (6.9277)
Ejecución 7	6.7160 (0.4704)	5.4406 (1.1410)	16.8850 (2.2979)	5.6924 (3.4497)	5.8222 (4.6138)	5.8211 (6.9137)
Ejecución 8	6.0101 (0.4663)	10.3748 (1.1530)	8.6158 (2.3010)	12.2407 (3.4613)	5.0171 (4.632)	13.3754 (6.9847)
Ejecución 9	11.4108 (0.4578)	5.6826 (1.1569)	16.2749 (2.2975)	5.3577 (3.4452)	5.9271 (4.6026)	11.3097 (6.9684)
Ejecución 10	6.1856 (0.4622)	12.6162 (1.1547)	11.9785 (2.3007)	6.0950 (3.4339)	11.3015 (4.6118)	6.6023 (6.8889)
Media	12.3015 (0.4665)	11.2927 (1.1521)	12.4489 (2.3082)	7.4194 (3.4469)	6.6338 (4.6120)	7.8001 (6.9217)

Observando los datos de la Tabla 5.4, representados en la Figura 5.6, puede confirmarse de nuevo tanto el aumento lineal que sufren los tiempos de ejecución como la tendencia a la disminución de los mejores valores de la función objetivo (lógico porque se está minimizando) con el aumento de población. Es cierto que la disminución se estabiliza tras un tamaño de 1000, pero los tiempos siguen aumentando. Puede verse que el valor medio de la función objetivo para un tamaño de 1500 individuos empeora al obtenido con 1000 individuos, lo cual no debería haber ocurrido puesto que lo esperable sería que se hubiera obtenido un valor parecido. No obstante, hay que recordar que se está tratando con algoritmos principalmente aleatorios y que puede darse el caso en el que en esas 10 iteraciones precisamente la población inicial que se generó no era muy adecuada.

También es preciso comentar que el comportamiento para tamaños de población pequeños es totalmente el esperado, ya que generan resultados parecidos al del algoritmo aleatorio programado. Esto se debe que al tener un espacio de búsqueda tan amplio y una función a optimizar tan compleja el tamaño de población debe ser lo suficientemente grande para abarcar gran parte de dicho espacio de búsqueda y poder evolucionar hacia el óptimo con el paso de las generaciones. Al utilizar tamaños pequeños, se exploran pocas zonas del espacio siendo menos probable el encontrar buenos resultados.

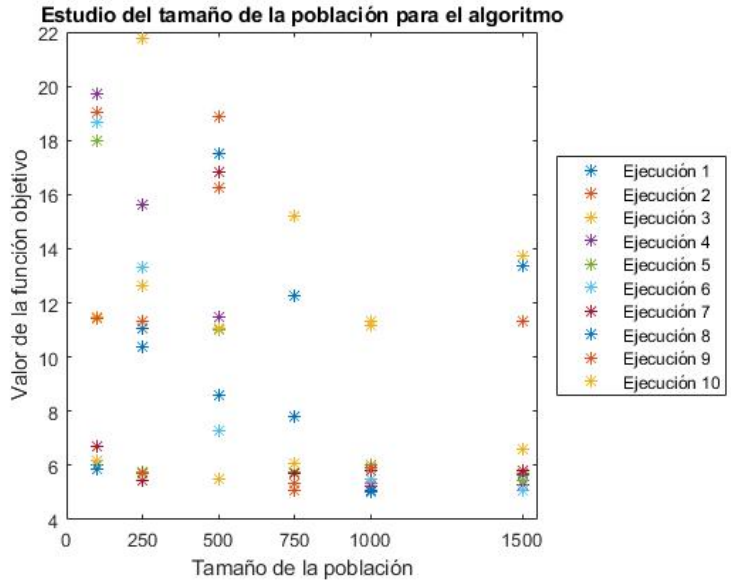


Figura 5.6 Representación gráfica de los valores obtenidos para las 10 ejecuciones.

En último lugar se comenta cuál será el tamaño de población escogido, parece razonable (viendo la Figura 5.7) que establecer un tamaño de 1000 individuos será lo ideal para este problema ya que permite encontrar soluciones aceptables en sólo 1000 generaciones para tiempos que rondan los 4 minutos y medio. Puesto que para la optimización final del problema se aumentará el límite de generaciones se conseguirán obtener aún mejores resultados.

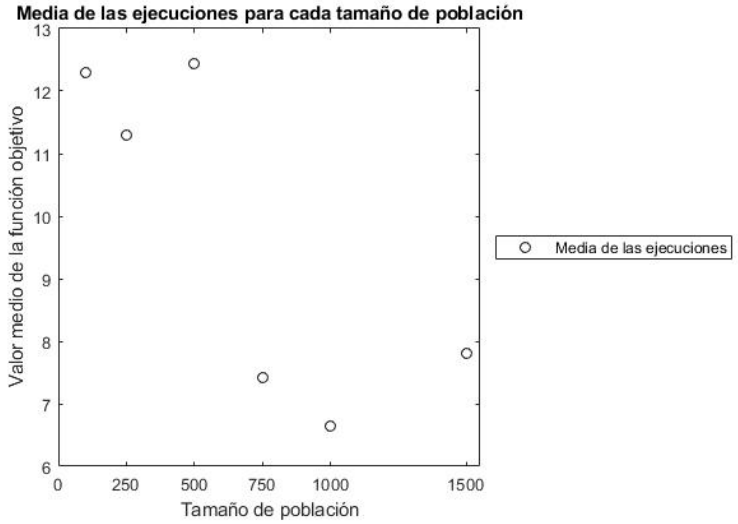


Figura 5.7 Representación gráfica de los valores medios obtenidos para las 10 ejecuciones.

5.1.4 Población élite

El número de individuos que pertenecerán a la élite, es decir, que no se verán afectados en el cambio de generación será el último parámetro a determinar en este análisis previo para la aplicación del algoritmo genético a los problemas sin maniobras en espacio profundo. Lo más adecuado al hablar de la población de élite es hacerlo en términos de porcentaje respecto a la población total, ya que dará una idea acerca de cómo de conservador es un algoritmo. Durante el análisis realizado para tres porcentajes diferentes, se han hecho uso de los parámetros anteriormente estudiados con sus valores adecuados:

- Factor de mutación: 0.02
- Parámetro de cruce: 0.75
- Tamaño de la población: 1000

Tras el, ya utilizado, análisis de 10 ejecuciones para 1000 generaciones como límite y de una duración de unas 2.4177 horas, se ha conseguido rellenar la Tabla 5.5 donde se muestran los resultados según la estructura: *Mejor valor de la función objetivo (Media de los valores de la función objetivo en la última generación)*.

Tabla 5.5 Resultados obtenidos del estudio de la población de la élite para el algoritmo genético con el problema *Cassini1*.

ÉLITE	1 %	5 %	10 %
Ejecución 1	11.3064 (11.4855)	5.1053 (5.2115)	5.7907 (5.7909)
Ejecución 2	5.7544 (6.8228)	6.0575 (6.0582)	5.5655 (5.5658)
Ejecución 3	5.3912 (5.9078)	5.8512 (5.9633)	5.4596 (5.4841)
Ejecución 4	6.1595 (6.1597)	5.9482 (6.0961)	7.1923 (7.2088)
Ejecución 5	5.1924 (6.3520)	5.8877 (6.1352)	12.9555 (12.9605)
Ejecución 6	5.6042 (7.0837)	11.3361 (11.4432)	7.1136 (7.1346)
Ejecución 7	5.6638 (5.9541)	5.4715 (5.5122)	5.3106 (5.3121)
Ejecución 8	5.3648 (5.7178)	6.7009 (7.0185)	6.4266 (6.5965)
Ejecución 9	11.1467 (11.9051)	11.2216 (11.4943)	5.4401 (5.4788)
Ejecución 10	5.5141 (5.5340)	5.3847 (5.4961)	11.6766 (11.6856)
Media	6.7095 (7.2923)	6.8965 (7.0429)	7.2931 (7.3218)
Diferencia	0.5828	0.1464	0.0287

Respecto a la penúltima fila, mencionar que vuelve a mostrar la media de las 10 ejecuciones tanto para el valor de la función objetivo como para las medias de las últimas generaciones. Y la última, muestra la diferencia entre los dos valores presentados en la penúltima fila a modo de desviación entre el mejor valor encontrado y la media del total de puntos encontrados en la última generación.

Estos valores de la última fila demuestran lo que cabía esperar y es que a medida que aumenta la cantidad de población que evoluciona sin sufrir modificaciones el algoritmo se vuelve más conservador. Por ello, en las nuevas generaciones comienzan a aparecer individuos muy parecidos a los ya existentes provocando que la media de todos los valores de la función objetivo que proporcionan se parezca mucho al mejor valor encontrado.

En cambio, si evolucionan pocos individuos sin modificaciones da lugar a una búsqueda más libre, lo cual permite obtener mejores valores finales de la función objetivo, de modo que la diferencia con la media del resto de valores que proporcionan los individuos se verá afectada. Puesto que

siempre se van a mantener a los mejores de cada generación, se consigue que el algoritmo nunca de como resultado de una generación un valor mayor a uno de una generación anterior. Se generará una curva de resultados descendiente o, a lo sumo, constante si no encuentra resultados mejores. Para las optimizaciones que van a realizarse se tomará que la población de élite tenga un tamaño del 1 % de la población total, puesto que es el que mejores resultados ha generado de la función objetivo. Además, porque interesa un cierto nivel de exploración debido a la complejidad de las funciones a estudiar.

5.2 Enjambre de partículas

Análogamente al algoritmo genético, el algoritmo de optimización por enjambre de partículas también dependerá de una serie de parámetros que favorecerán una buena actuación si se escogen adecuadamente. Dichos parámetros son: el factor de inercia, el coeficiente de atracción personal, el coeficiente de atracción global y el tamaño del enjambre. El análisis se ha dividido en dos partes: inicialmente se han estudiado una serie de combinaciones de los tres parámetros cinemáticos y, una vez fijados, se ha investigado acerca del tamaño de enjambre óptimo para los problemas sin maniobras en espacio profundo.

5.2.1 Parámetros cinemáticos: factor de inercia y coeficientes de atracción

Debido a la complejidad que tiene el analizar diversas combinaciones de tres factores que no tienen ninguna limitación, se ha tenido que imponer una simplificación que permitirá, además, saber cuánto afecta cada parámetro a la hora de actualizar las velocidades de cada partícula. Esta simplificación es simple y trata de establecer que la suma de los tres coeficientes sea igual a un cierto número natural. Como se verá a continuación, el análisis de los coeficientes se divide a su vez en cuatro partes diferenciadas: presentación de las diferentes configuraciones, estudio de dichas configuraciones para el caso de que los coeficientes sumen 2, estudio para el caso en el que sumen 1 o 3 y finalmente se muestra un análisis de algunos casos especiales.

Configuraciones analizadas

Las configuraciones escogidas son algunas entre tantas otras, puesto que existen infinitas que sigan cumpliendo con la única condición que se tiene: la suma de los tres coeficientes debe valer 2. Se ha intentado escoger valores sencillos que den lugar a unos porcentajes sobre el total fáciles de reconocer. En la Tabla 5.6 se exponen los coeficientes de las diferentes configuraciones a estudiar y, entre paréntesis, sus porcentajes respecto de la suma total.

Tabla 5.6 Configuraciones escogidas para analizar los posibles valores de los parámetros cinemáticos del algoritmo.

CONFIGURACIONES	Inercia	Atrac. Personal	Atrac. Global
C1	1 (50 %)	0.5 (25 %)	0.5 (25 %)
C2	1 (50 %)	0.25 (12.5 %)	0.75 (37.5 %)
C3	1 (50 %)	0.75 (37.5 %)	0.25 (12.5 %)
C4	0.5 (25 %)	0.75 (37.5 %)	0.75 (37.5 %)
C5	0.5 (25 %)	0.5 (25 %)	1 (50 %)
C6	0.5 (25 %)	1 (50 %)	0.5 (25 %)
C7	0.25 (12.5 %)	0.875 (43.75 %)	0.875 (43.75 %)
C8	0.25 (12.5 %)	0.5 (25 %)	1.25 (62.5 %)
C9	0.25 (12.5 %)	1.25 (62.5 %)	0.5 (25 %)
C10	0.667 (33.4 %)	0.666 (33.3 %)	0.666 (33.3 %)

Primer estudio de las configuraciones escogidas

En este caso, como ya se ha expuesto, todas las configuraciones se han generado de forma que la suma de los tres coeficientes sea 2. El estudio realizado consta de 10 ejecuciones que proporcionarán la suficiente información como para escoger una de las configuraciones de la Tabla 5.6. Al igual que para el algoritmo genético, el parámetro no estudiado en este primer análisis, el tamaño del enjambre, deberá fijarse siguiendo algunas ejecuciones de prueba realizadas para entender tanto el algoritmo como los problemas a optimizar. Gracias a esas primeras pruebas se fija el tamaño en:

- Tamaño del enjambre: 1000

Como limitación para este análisis se tiene el número máximo de iteraciones a realizar, que se establece en 1000. Este valor es suficiente para analizar el comportamiento del algoritmo en las primeras iteraciones y las tendencias de los resultados según la configuración estudiada. La duración de este análisis ha sido de 9.829 horas y ha resultado en la Tabla 5.7.

Tabla 5.7 Resultados obtenidos del estudio de las configuraciones para el algoritmo de optimización por enjambre de partículas con el problema *Cassini1*.

CONFI. PROP.	C1	C2	C3	C4	C5
Ejecución 1	5.7849	5.9821	5.6917	5.3179	5.9028
Ejecución 2	12.7466	12.7832	5.8625	5.3110	5.8552
Ejecución 3	5.8621	6.2099	5.6450	11.8665	5.3098
Ejecución 4	13.3274	6.3567	6.2429	5.2395	12.6714
Ejecución 5	5.5876	5.8873	5.9721	5.6100	13.1290
Ejecución 6	12.8395	12.1102	5.5890	11.8384	10.9981
Ejecución 7	5.7258	12.1278	5.7350	11.3428	5.3107
Ejecución 8	12.7421	13.0155	6.3149	5.8917	11.0351
Ejecución 9	5.5752	12.8803	5.7029	11.0718	12.6166
Ejecución 10	5.9144	5.9496	5.8974	5.7612	11.3128
Media	8.6106	9.3303	5.8654	7.4126	9.4131

CONFI. PROP.	C6	C7	C8	C9	C10
Ejecución 1	5.9513	5.8319	11.0739	6.1788	6.0699
Ejecución 2	11.1856	5.3645	12.6047	9.7217	11.0347
Ejecución 3	6.2495	10.2975	5.3471	6.4828	11.0531
Ejecución 4	6.0983	12.5959	5.3213	5.8906	6.2843
Ejecución 5	5.8032	11.2182	10.9965	6.1090	5.3115
Ejecución 6	5.3705	11.0171	11.1498	9.7176	5.3352
Ejecución 7	11.0039	11.1103	12.5895	6.1530	11.0649
Ejecución 8	5.7759	11.0771	11.0740	6.1037	5.3106
Ejecución 9	5.9026	6.0063	11.0533	11.2230	5.3161
Ejecución 10	5.9991	5.4307	11.0343	6.2877	11.0604
Media	6.9340	8.9950	10.2244	7.3868	7.7841

Si bien no parece que pueda llegarse a ninguna conclusión, si se realiza un repaso general de la Tabla 5.7, cuyos valores se han representado en la Figura 5.8, puede verse que si se agrupan las configuraciones según el valor del factor de inercia, es decir, en tríos ({C1 C2 C3}, {C4 C5 C6} y {C7 C8 C9}), existe un cierto patrón en los resultados. Este patrón suele darse casi para todas las ejecuciones y es el que sigue: para las primeras componentes de los tríos (donde ambos parámetros

de atracción son idénticos) se obtienen unos ciertos valores generalmente altos respecto al óptimo buscado; para las segundas componentes (donde el coeficiente de atracción global es mayor) se consiguen valores aún peores que los anteriores; y para las terceras componentes (donde el coeficiente de atracción personal es mayor) se alcanzan valores más cercanos al óptimo del problema (4.9307).

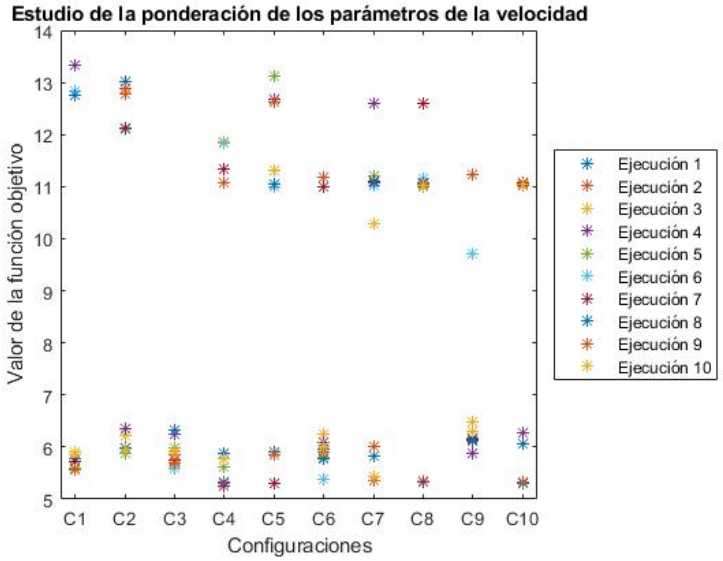


Figura 5.8 Representación gráfica de los valores obtenidos para las 10 ejecuciones.

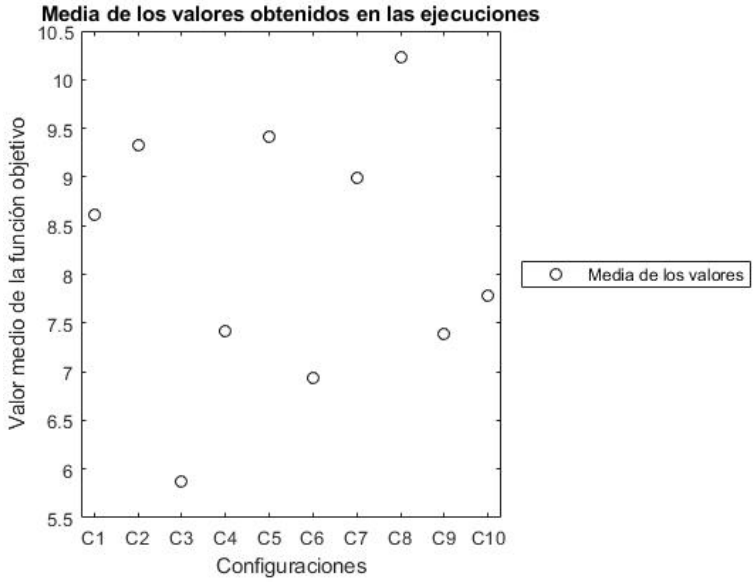


Figura 5.9 Representación gráfica de los valores medios obtenidos para las 10 ejecuciones.

Este patrón tiene su explicación en la forma en la que se mueven las partículas en cada caso. Si el coeficiente de atracción global es mucho mayor que el personal, las partículas tenderán a alejarse del mejor resultado que han encontrado en busca del mejor encontrado globalmente, luego se tiene un algoritmo con poca explotación, o sea, más parecido a un algoritmo aleatorio (véase las similitudes de los resultados de las columnas C2, C4 y C8 con los del algoritmo aleatorio de la Tabla 5.2). Sin

embargo, en el caso de que el coeficiente de atracción personal sea mayor que el global el algoritmo tenderá principalmente a explotar las zonas de alrededor del mejor valor que ha encontrado en busca de otro mejor. Y, además, tendrá una cierta influencia del mejor valor global y de su inercia lo cual dota al algoritmo de una capacidad de exploración para evitar el estancamiento en un óptimo local.

Por todo lo desarrollado, parece que la opción más razonable es escoger una configuración donde el coeficiente de atracción personal sea mayor que el global. Adicionalmente, utilizando las medias de los mejores valores obtenidos, representados en la Figura 5.9, se concluye que la configuración a utilizar es la C3, por ser la que mejores valores ha proporcionado.

Segundo estudio de las configuraciones escogidas

Una vez elegida la configuración C3 y obteniendo de la Tabla 5.6 las proporciones de cada uno que van a utilizarse respecto a un cierto total:

$$\omega = 50\,\%, \quad c_1 = 37.5\,\% \text{ y } c_2 = 12.5\,\%,$$

se está en condiciones de realizar el estudio donde se cambia el valor de dicha suma total. En este caso, se mantiene el tamaño de enjambre utilizado anteriormente:

- Tamaño del enjambre: 1000

Tampoco se va a modificar el número máximo de actualizaciones de posición que ejecute el algoritmo, por tanto, realizará 1000 iteraciones. Definidos estos parámetros podrá realizarse el análisis que se pretende. La idea es comparar los casos en los que los tres coeficientes sumen los valores 1 y 3 para comparar los resultados que se obtienen con los conseguidos cuando suman 2. La variación de esta suma total afecta directamente a la actualización de la velocidad de las partículas y, por ende, a su posición. Cuanto mayor sea la suma de los tres coeficientes, puesto que las proporciones se van a mantener en las propuestas para la C3, mayores serán cada uno de los coeficientes y cuando se actualice la velocidad lo hará sumando un término de mayor magnitud por lo que variará más bruscamente. Los valores utilizados para cada uno de los coeficientes, según su suma, se presentan en la Tabla 5.8.

Tabla 5.8 Valor de los parámetros de la C3 según el valor de la suma total.

C3	ω	c_1	c_2
Suman 1	0.5	0.375	0.125
Suman 2	1	0.75	0.25
Suman 3	1.5	1.125	0.375

Realizando un análisis con 10 ejecuciones para cada caso, tras 2.9487 horas, el algoritmo ha proporcionado los valores que completan la Tabla 5.9.

Tabla 5.9 Resultados obtenidos del estudio de la configuración C3 para el algoritmo de optimización por enjambre de partículas con el problema *Cassini1*.

SUMA TOTAL	SOBRE 1	SOBRE 2	SOBRE 3
Ejecución 1	9.9619	5.6917	9.6132
Ejecución 2	8.6657	5.8625	14.2875
Ejecución 3	12.6126	5.6450	7.6681
Ejecución 4	5.7909	6.2429	9.2777
Ejecución 5	15.3223	5.9721	13.6562
Ejecución 6	15.9599	5.5890	13.6070
Ejecución 7	15.1871	5.7350	14.1734
Ejecución 8	12.4835	6.3149	14.3311
Ejecución 9	10.9514	5.7029	8.1201
Ejecución 10	7.6358	5.8974	7.0430
Media	10.6935	5.8654	11.1777

Observando los resultados, representados en la Figura 5.10, es evidente la diferencia que hay entre el funcionamiento del algoritmo cuando suman 2 y el funcionamiento en los otros casos. La explicación a estos resultados se basa en la actualización de las velocidades de cada partícula. Para el caso en el que suman 1, los tres coeficientes son pequeños y, por tanto, las nuevas velocidades que aparecen son de pequeña magnitud. Esto provoca que la nueva posición en la que se sitúa una partícula tras la iteración sea cercana a la posición desde la que parte, no permitiendo entonces que las partículas abarquen y se muevan por un gran espacio de búsqueda. Se tiene un algoritmo de explotación de zonas, útil en el caso de conocer que el óptimo del problema se encuentra en una zona definida.

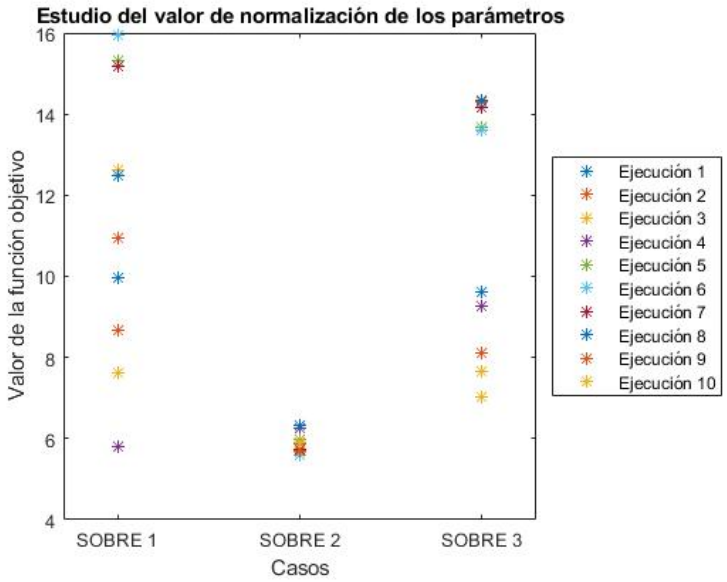


Figura 5.10 Representación gráfica de los valores obtenidos para las 10 ejecuciones.

Por otra parte, si la suma es 3, ocurre lo opuesto, los coeficientes serán grandes y las velocidades que actualizan las posiciones también serán grandes, así que la posición a la que se llega tras la iteración estará alejada de la posición anterior tendiéndose a un algoritmo aleatorio cuanto mayor sean dichos coeficientes. Véase que los resultados de la cuarta columna de la Tabla 5.9 son parecidos a los obtenidos por el algoritmo aleatorio.

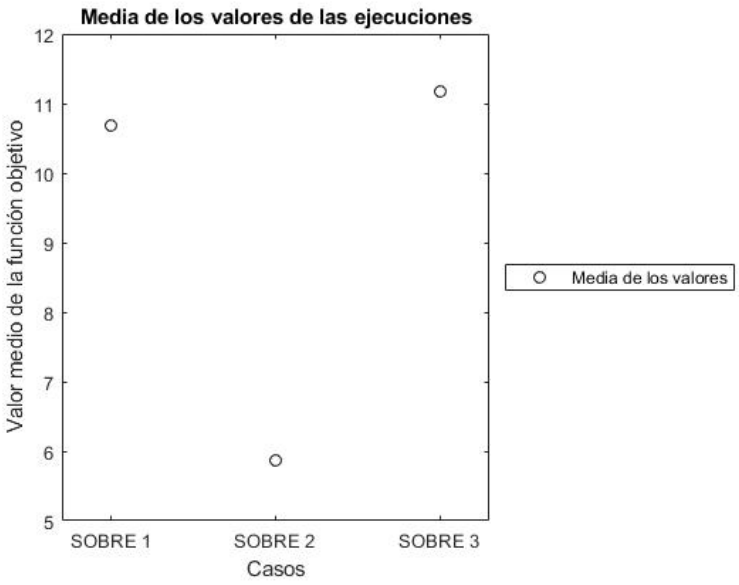


Figura 5.11 Representación gráfica de los valores medios obtenidos para las 10 ejecuciones.

Como conclusión del análisis, teniendo en cuenta los valores medios obtenidos, y representados en la Figura 5.11, se elegirá la configuración C3 de tal forma que sus coeficientes sumen un total de 2.

Estudio de casos especiales

Para completar el estudio de los parámetros cinemáticos que afectan a la evolución de las posiciones de las diferentes partículas del enjambre, se han analizado también algunos casos especiales recogidos en la Tabla 5.10.

Tabla 5.10 Configuraciones especiales escogidas para analizar.

CONFIGURACIONES	Inercia	Atrac. Personal	Atrac. Global
D1	0	0	2 (100%)
D2	0	2 (100%)	0
D3	0	1 (50%)	1 (50%)
D4	2 (100%)	0	0
D5	1 (50%)	0	1 (50%)
D6	1 (50%)	1 (50%)	0

Los casos a destacar son: D1, solamente tiene en cuenta para la actualización de la posición el mejor global; D2, todos los puntos tienden a su mejor valor (permanecerán estáticos); y D4, la nueva velocidad tiene la misma dirección que la velocidad con la que se mueve (movimiento rectilíneo de las partículas). Si se ejecuta cada caso 10 veces y se recogen los resultados, tras un tiempo de 5.8974 horas se habrá rellenado la Tabla 5.11. Como en el resto de análisis de configuraciones se ha mantenido un tamaño de enjambre de 1000 partículas y un límite de 1000 actualizaciones de posición.

Tabla 5.11 Resultados obtenidos del estudio de las configuraciones especiales para el algoritmo de optimización por enjambre de partículas con el problema *Cassini1*.

CONFIGURACIÓN	D1	D2	D3	D4	D5	D6
Ejecución 1	11.1201	27.3677	11.2749	11.5923	5.8697	9.0681
Ejecución 2	6.5231	28.8297	11.4922	40.0659	13.4089	7.7249
Ejecución 3	11.1055	37.7838	11.1036	32.0938	13.0422	9.6118
Ejecución 4	11.2190	31.3235	11.2063	21.8634	12.3929	7.9597
Ejecución 5	13.3297	49.0046	5.8854	20.3526	6.2765	7.8948
Ejecución 6	8.9084	30.6538	14.1097	29.6028	13.4597	7.2398
Ejecución 7	14.2852	29.0791	11.2601	18.0166	5.9579	8.8913
Ejecución 8	11.0776	37.4727	11.4886	24.5900	5.9935	7.2739
Ejecución 9	11.5912	24.8344	6.2723	31.5338	12.9725	7.9981
Ejecución 10	8.8198	34.2075	11.2882	29.3695	5.9718	8.3709
Media	10.7980	33.0557	10.5381	25.9081	9.5346	8.2033

En cuanto al análisis de los resultados de la Tabla 5.11, representados en la Figura 5.12, se van a tratar los seis casos estudiados por separado para que sea posible comentar sus peculiaridades.

Comenzando por la configuración D1, donde el único coeficiente presente en la actualización de las velocidades es el de atracción global, puede asimilarse a una situación donde existiese un sumidero colocado en la posición del mejor valor de la función objetivo para las posiciones iniciales y el resto de partículas realizan un movimiento rectilíneo hacia dicho sumidero con el paso de las iteraciones. Puesto que las partículas se irán moviendo por el espacio de búsqueda según se acercan a dicho sumidero, es posible que alguna de ellas encuentre un mejor valor durante el trayecto por lo que, en ese caso, el sumidero cambiaría su posición a ese nuevo punto que proporciona un mejor valor de la función objetivo y el resto de partículas cambiarían su foco de atracción a ese nuevo lugar. Por esa capacidad de mejora, en la columna de resultados de D1 existe algún punto con un valor de la función objetivo bajo, pero lejos del óptimo.

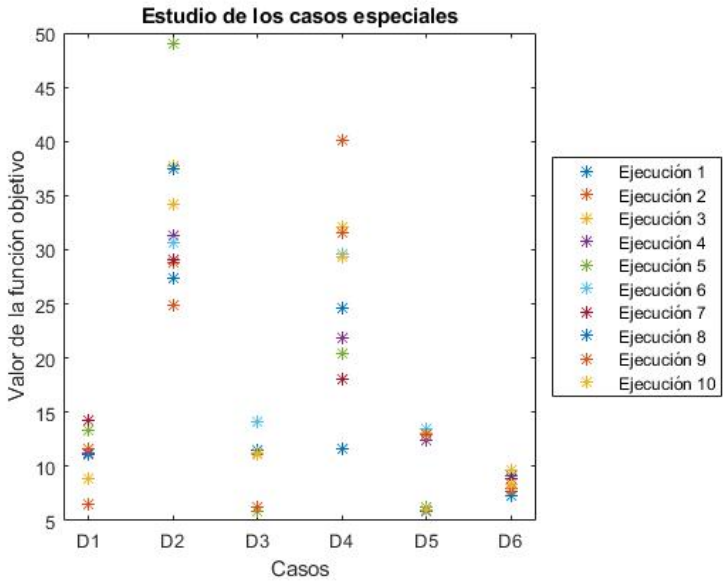


Figura 5.12 Representación gráfica de los valores obtenidos para las 10 ejecuciones.

En la configuración D2 las partículas permanecen estáticas, luego tiene un comportamiento idéntico a un algoritmo aleatorio en el que se toman 1000 puntos cualesquiera de todo el espacio de búsqueda. La estaticidad de las partículas se debe a que únicamente se ven atraídas por el mejor valor encontrado por ellas mismas y debido a que dicho valor para la primera iteración resulta del propio punto en el que se generan, no se moverán de la posición de partida. Puede verse que los resultados son nefastos por la falta de evolución de algoritmo.

Para la configuración D3, se tiene un comportamiento parecido al de D1 pero con el lastre de que las partículas también están atraídas por el mejor personal, luego las trayectorias dejan de ser rectilíneas. Se obtienen resultados parecidos a D1.

Respecto a la configuración D4, es destacable que de nuevo vuelve a tenerse un movimiento rectilíneo de las partículas, pero ahora sin tender hacia ningún punto en concreto. La dirección y sentido del movimiento de cada partícula estará definido por la velocidad inicial aleatoria que le asigna el propio algoritmo a cada partícula, es decir, una vez fijada dicha velocidad, la partícula se moverá en la misma dirección hasta que la posición de la partícula alcance los límites permitidos. Como consecuencia de ser independiente de los mejores puntos encontrados, el algoritmo no explota las zonas cercanas a dichos puntos en busca de otros mejores y solamente encuentra otros resultados si por suerte una partícula pasa por alguna posición mejor. Por norma general, los resultados obtenidos serán malos.

Finalmente, englobando las configuraciones D5 y D6, los mejores resultados se obtienen cuando se combinan tanto el efecto de la inercia de la partícula como los de atracción hacia los mejores resultados. En estos, cada partícula seguirá una trayectoria caótica pero con una cierta tendencia a explorar zonas en las que es posible encontrar mejores resultados. Al igual que ocurrió para las configuraciones de la Tabla 5.6, el caso en el que el coeficiente de atracción personal es mayor que el global los resultados están más cerca del óptimo generalmente (véase la Figura 5.13).

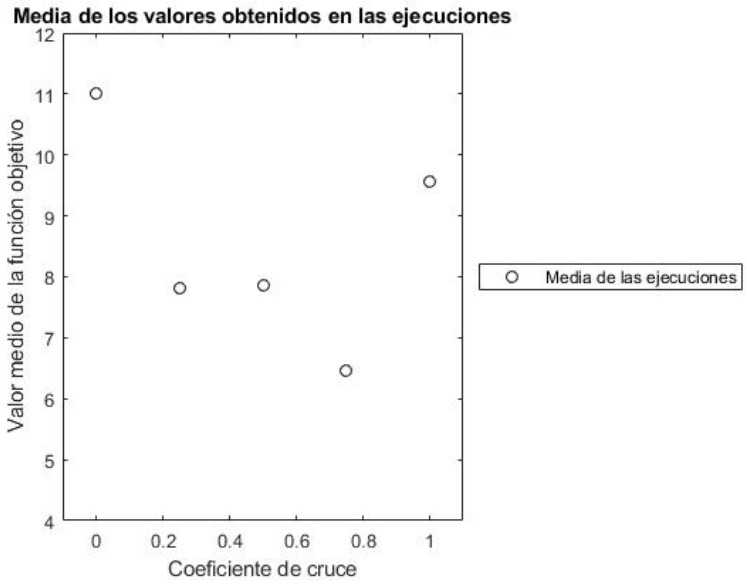


Figura 5.13 Representación gráfica de los valores medios obtenidos para las 10 ejecuciones.

Como conclusión final, recordar que este estudio se ha realizado a modo de complemento del estudio real de los parámetros cinemáticos para mostrar qué ocurre si se anula algunos de ellos.

Tras analizar los resultados, se deduce que no es una opción adecuada utilizar estas configuraciones especiales si se quiere optimizar una función compleja.

5.2.2 Tamaño del enjambre

El tamaño del enjambre, como ocurre para el tamaño de la población del algoritmo genético, es el parámetro que facilita en mayor medida la búsqueda del óptimo de una función. Es directamente proporcional la relación que existe entre el tamaño del enjambre y la probabilidad de encontrar el punto óptimo, ya que cuantas más partículas se estén considerando, más espacio se estará explorando en cada iteración. La limitación que se tiene a la hora de elegir un cierto tamaño es el tiempo de ejecución puesto que en el caso de este algoritmo el crecimiento tiene un carácter exponencial según aumenta el número de partículas.

Para este análisis se han introducido los parámetros cinemáticos definidos anteriormente, lo cual permitirá comparar el funcionamiento del algoritmo según el tamaño del enjambre con unas condiciones de actualización de velocidad y posición adecuadas. Los valores utilizados son:

- Factor de inercia: 1
- Coeficiente de atracción del mejor personal: 0.75
- Coeficiente de atracción del mejor global: 0.25

Puesto que van a realizarse 10 ejecuciones para cada tamaño de población escogido, se mantendrá el límite de 1000 generaciones para que el tiempo de duración del análisis sea razonable. En cuanto a los tamaños de enjambre escogidos se ha seguido las recomendaciones realizadas por los investigadores polacos A. P. Piotrowski, J. J. Napiorkowski y A. E. Piotrowska en su artículo [9] y, además, se han incluido los tamaños de 1000 y 1500 partículas ya que se está analizando una función compleja y es posible que generen mejores resultados. Transcurridas 3.6833 horas de cálculo computacional, se consigue la Tabla 5.12 de resultados. Los tiempos medios de cada ejecución según el tamaño se muestran en la Tabla 5.13.

Tabla 5.12 Resultados obtenidos del estudio del tamaño del enjambre para el algoritmo de optimización por enjambre de partículas con el problema *Cassini1*.

TAMAÑO	50	100	300	500	700	1000	1500
Ejecución 1	14.1544	6.5990	12.8600	12.8947	5.6248	5.6917	6.6961
Ejecución 2	13.1529	7.2970	12.9086	12.7233	6.0943	5.8625	5.7133
Ejecución 3	13.7712	13.4103	5.6370	5.6735	5.6918	5.6450	5.4853
Ejecución 4	13.7844	5.9920	6.0647	6.0733	12.9287	6.2429	5.4786
Ejecución 5	12.4483	13.0974	10.5685	10.0980	12.7908	5.9721	5.6541
Ejecución 6	13.5206	6.4015	6.0358	5.8139	5.9344	5.5890	9.5555
Ejecución 7	13.3761	6.5124	6.5944	6.0107	5.8616	5.7350	12.8253
Ejecución 8	13.5581	6.7110	6.0411	12.0599	6.4462	6.3149	5.5775
Ejecución 9	14.1390	6.6227	6.3864	6.1289	6.2174	5.7029	5.7173
Ejecución 10	7.8598	6.3233	10.5840	6.3292	5.6419	5.8974	5.6494
Media	12.9755	7.8967	8.3681	8.3799	7.3232	5.8654	6.8352

Tabla 5.13 Tiempos medios por ejecución según el tamaño del enjambre.

TAMAÑO	TIEMPO (min)
50	0.2331
100	0.4679
300	1.4318
500	2.4769
700	3.5645
1000	5.3325
1500	8.5932

Si se revisan los resultados expuestos, y representados en la Figura 5.14, se deduce de manera sencilla como a medida que crece el tamaño del enjambre se van obteniendo valores más cercanos al óptimo del problema hasta un cierto punto. A partir de 1000 partículas, el valor medio de los mejores resultados de la función objetivo crece aunque debe considerarse que se estabiliza el comportamiento. Es decir, a partir de 1000 partículas el tamaño no influye tanto a la hora de encontrar mejores resultados pero sí que incrementa el tiempo considerablemente (véase este comportamiento en la Figura 5.15).

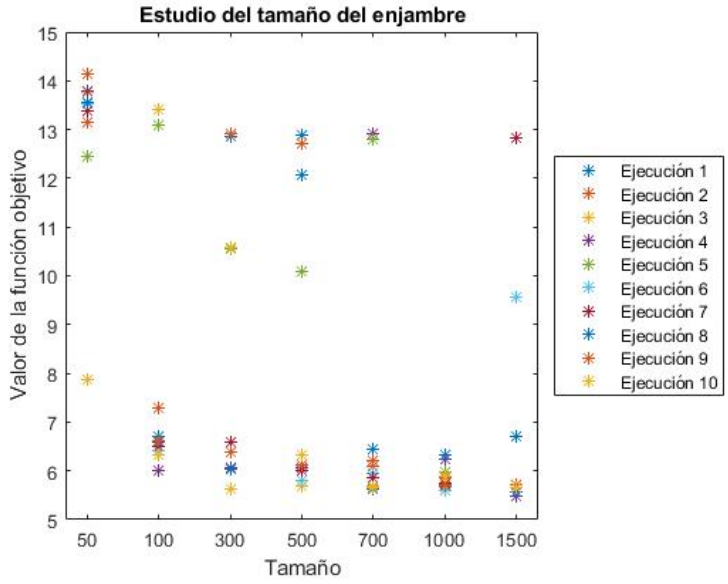


Figura 5.14 Representación gráfica de los valores obtenidos para las 10 ejecuciones.

En los casos en los que se tienen pocas partículas, el algoritmo es incapaz de llegar a resultados cercanos al óptimo puesto que no puede explorar adecuadamente la función compleja que se está analizando.

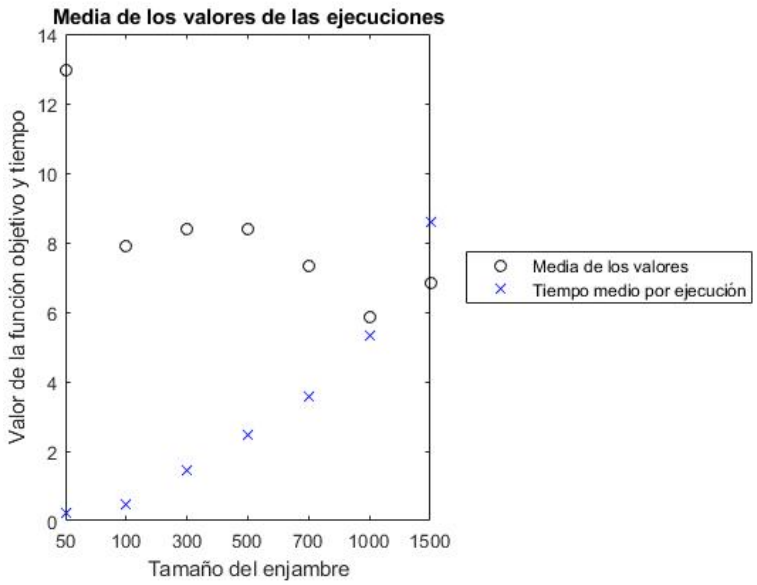


Figura 5.15 Representación gráfica de los valores medios obtenidos para las 10 ejecuciones.

Finalmente, tras comprobar y examinar los resultados, se infiere que el tamaño más adecuado para la optimización de los problemas es el de 1000 partículas, debido al equilibrio que proporciona en cuanto a resultados obtenidos y tiempos de ejecución.

6 Aplicación de algoritmos a los problemas

En este capítulo van a presentarse los resultados de las optimizaciones realizadas para los tres problemas considerados: *Cassini1*, *GTOC1* y *Messenger*. Para la aplicación de los diferentes algoritmos se han utilizado únicamente los parámetros que dan mejores resultados para cada problemas, ya que han sido analizados en el Capítulo 5. Junto a los dos algoritmos ya presentados, se ha aplicado otro que combina ambos, es decir, ejecuta un número N de generaciones con el algoritmo genético y sus resultados se proporcionan como partículas iniciales para el algoritmo de optimización por enjambre de partículas que realizará otras M actualizaciones de posición y sus resultados son la entrada de la siguiente ejecución del algoritmo genético. Es decir, se ha programado un algoritmo que secuencia el funcionamiento de los dos métodos de optimización. La idea de introducir esta novedad se basa en el intento de mejorar el funcionamiento de ambos algoritmos por separado.

El capítulo está dividido en dos secciones: la primera incluye los resultados de las optimizaciones de los problemas que no tienen maniobras en espacio profundo y la segunda recoge los resultados del problema con maniobras en espacio profundo.

6.1 Problemas sin maniobras en espacio profundo

6.1.1 Cassini1

Como ya se presentó en el Capítulo 4, el problema *Cassini1* trata de minimizar la suma total de impulsos a realizar en el viaje de una nave desde la Tierra hasta Saturno pasando por algunos planetas. Para la optimización se tiene un vector decisión de 6 componentes, que incluye el instante de partida y los tiempos de vuelos durante los trayectos heliocéntricos que se tienen.

Puesto que se ha optimizado la misión por tres métodos diferentes van a presentarse los resultados en tres secciones diferentes: una para el algoritmo genético, otra para el optimizador por enjambre de partículas y una última donde se recogen los resultados generados por el método mixto.

El objetivo del análisis de los parámetros del Capítulo 5 es conseguir que cada algoritmo tenga el mejor comportamiento posible adaptado al problema que se está resolviendo. Gracias a esto, con pocas generaciones (o iteraciones) del algoritmo y en tiempos relativamente pequeños se consiguen resultados bastante parecidos a los mejores publicados en la página de la ESA [4].

Para la demostración de que los algoritmos funcionan correctamente, sin la necesidad de estar días ejecutándose, se realizan 10 ejecuciones independientes de cada algoritmo para comprobar el nivel de calidad de los resultados obtenidos comparados con los publicados.

Algoritmo genético

A modo de resumen, van a comentarse los parámetros utilizados en la aplicación del algoritmo genético, que no son otros que los hallados en el Capítulo 5. Estos parámetros son:

- Factor de mutación: 0.02
- Parámetro de cruce: 0.75
- Tamaño de la población: 1000 individuos
- Población de la élite: 1 % del total de la población

Respecto a la condición de parada se ha utilizado únicamente el límite para las generaciones, imponiéndose un número máximo de 7500 generaciones. Este valor se ha escogido empíricamente, es decir, durante todas las ejecuciones realizadas desde el inicio del estudio del problema se ha comprobado que, por norma general, cuando el algoritmo llega a 7500 generaciones ya se ha estancado en una cierta solución (un mínimo local típicamente). Esto no elimina la posibilidad de que haya ejecuciones en las que fuera posible seguir mejorando los resultados tras las 7500 generaciones, pero se recuerda que el objetivo es encontrar la mejor solución posible en un tiempo pequeño.

En lo que sigue se exponen diez figuras correspondientes a las diez ejecuciones realizadas donde se muestra en puntos negros la evolución del mejor valor encontrado y en puntos azules los valores de la media de cada generación. Tras estas figuras, se añade una tabla que recoge los resultados.

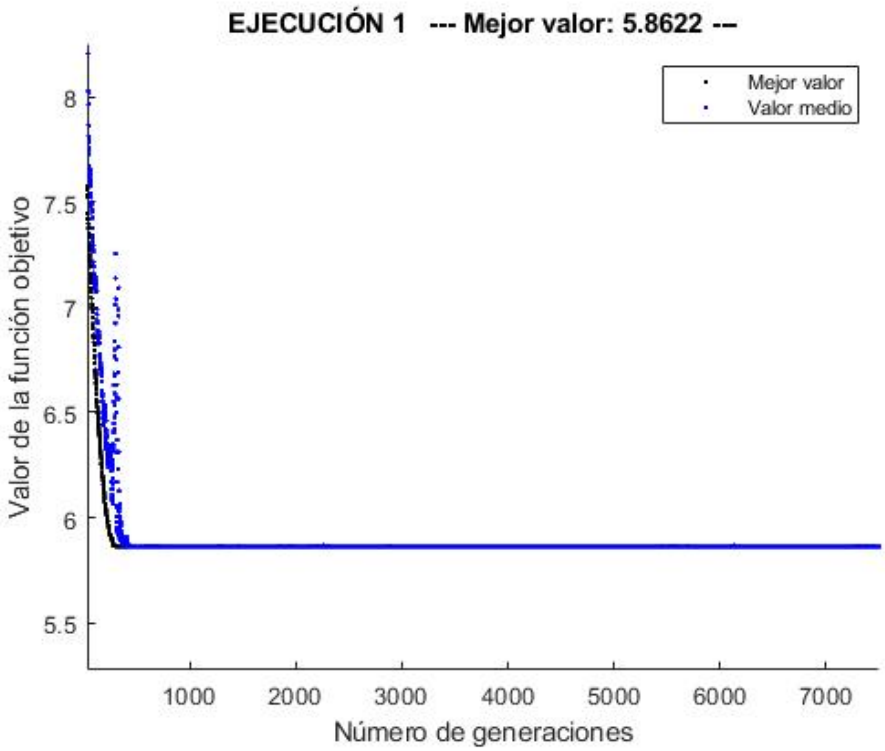


Figura 6.1 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la primera ejecución del algoritmo genético en el problema *Cassini1*.

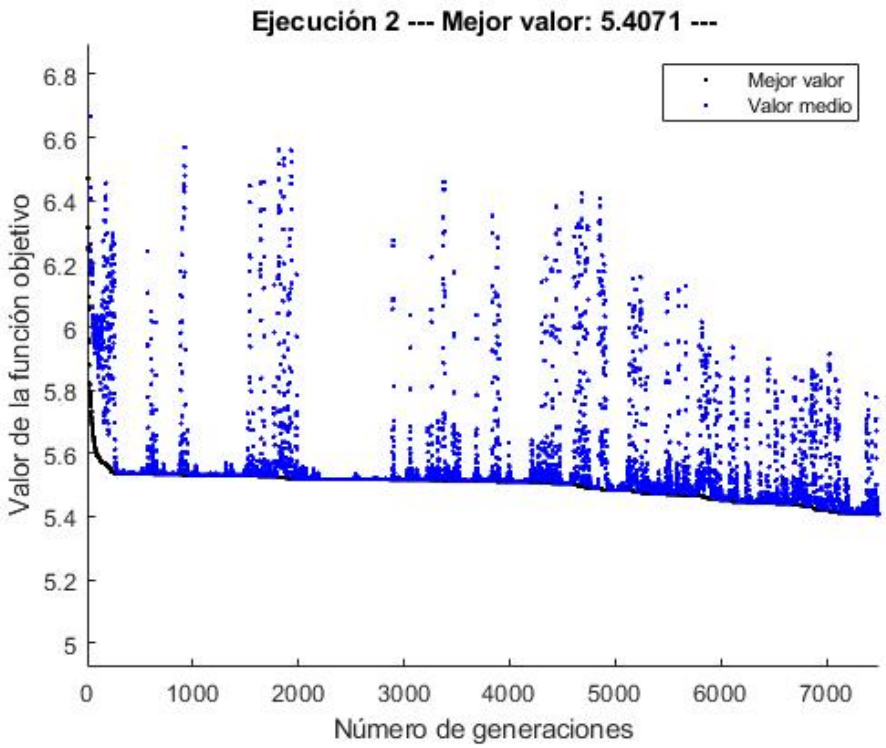


Figura 6.2 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la segunda ejecución del algoritmo genético en el problema *Cassini1*.

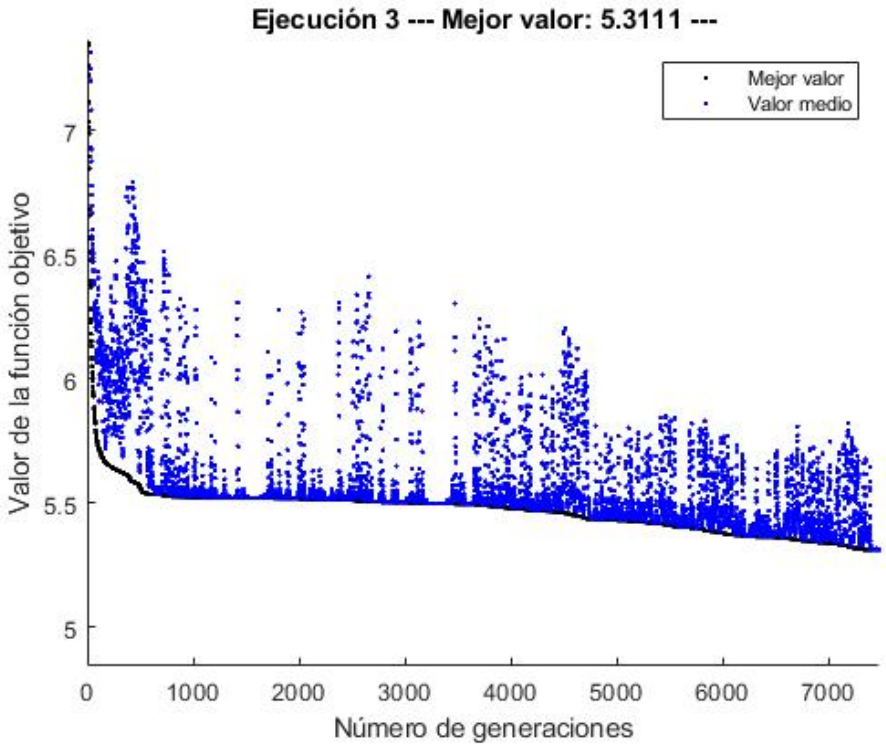


Figura 6.3 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la tercera ejecución del algoritmo genético en el problema *Cassini1*.

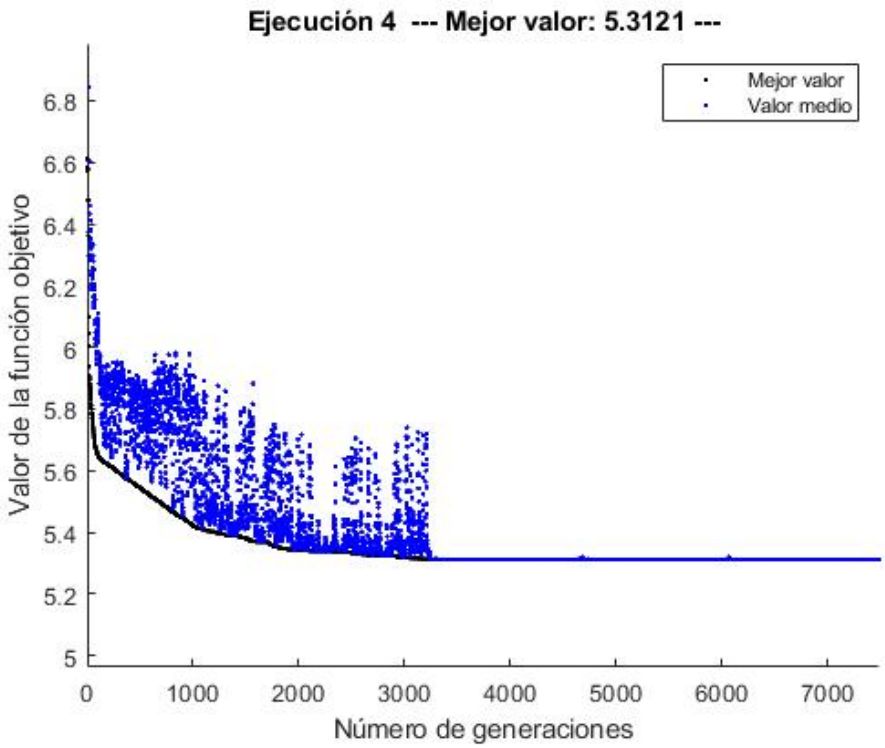


Figura 6.4 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la cuarta ejecución del algoritmo genético en el problema *Cassini1*.

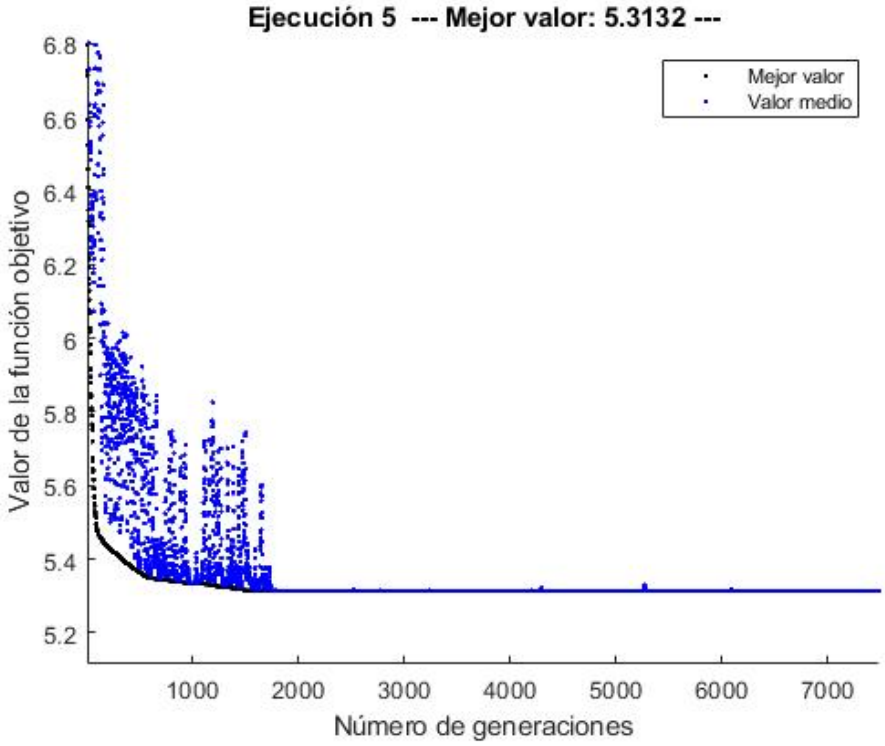


Figura 6.5 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la quinta ejecución del algoritmo genético en el problema *Cassini1*.

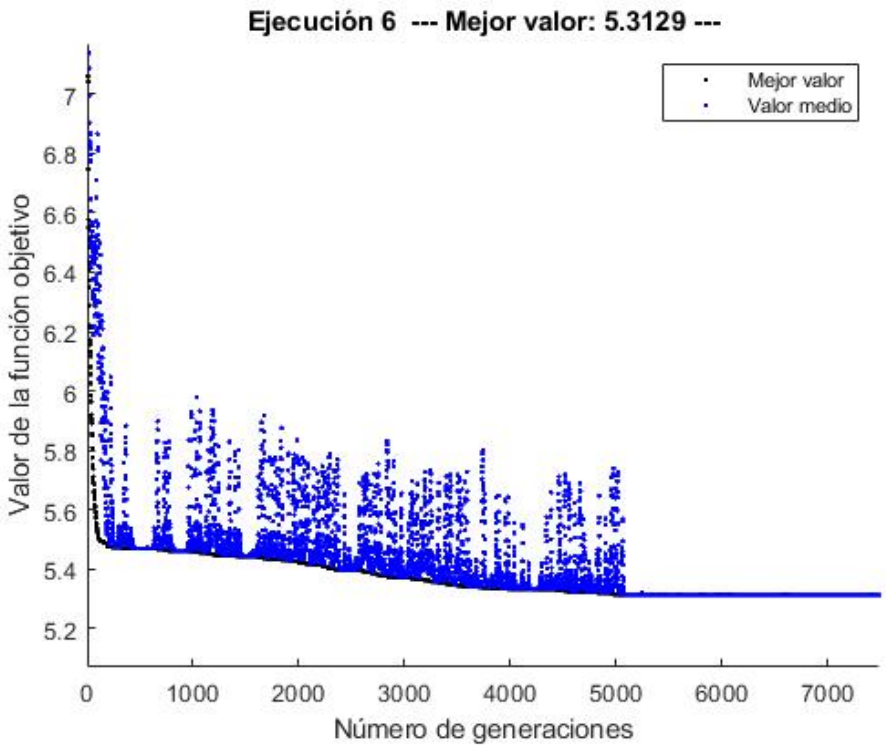


Figura 6.6 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la sexta ejecución del algoritmo genético en el problema *Cassini1*.

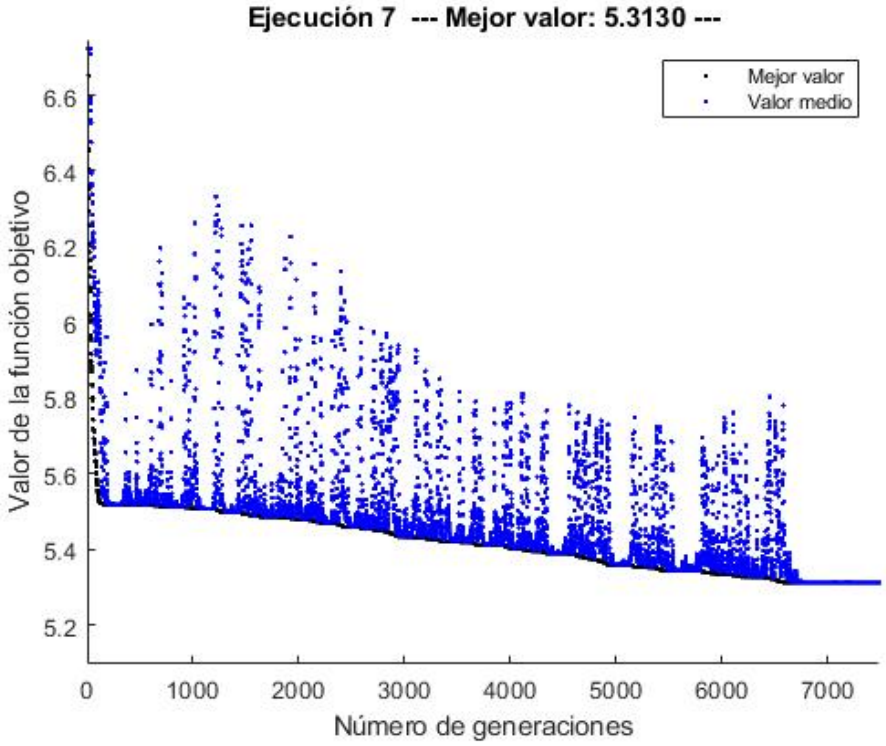


Figura 6.7 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la séptima ejecución del algoritmo genético en el problema *Cassini1*.

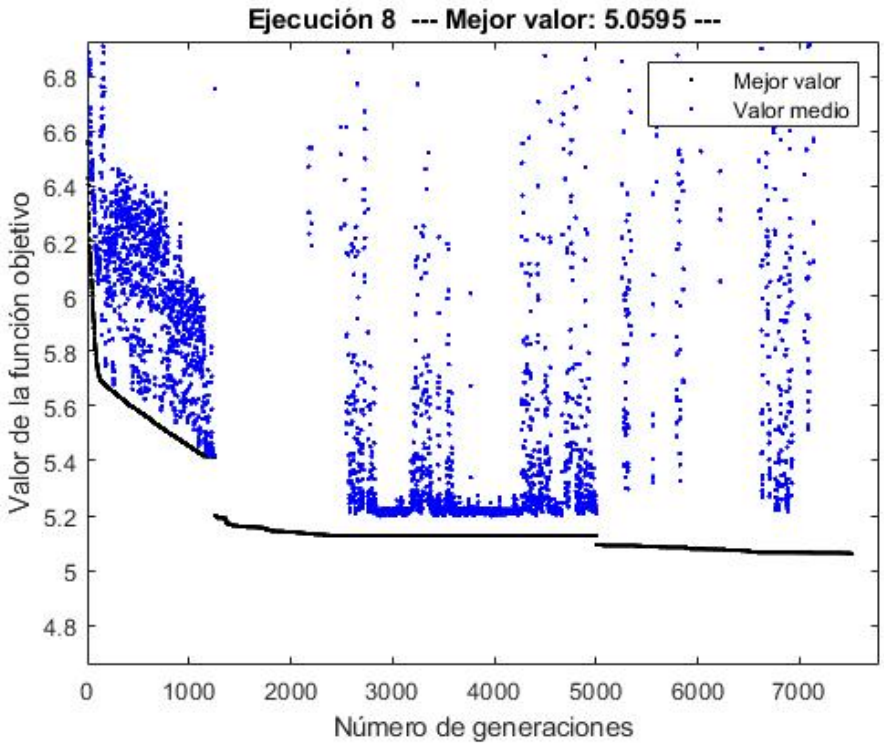


Figura 6.8 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la octava ejecución del algoritmo genético en el problema *Cassini1*.

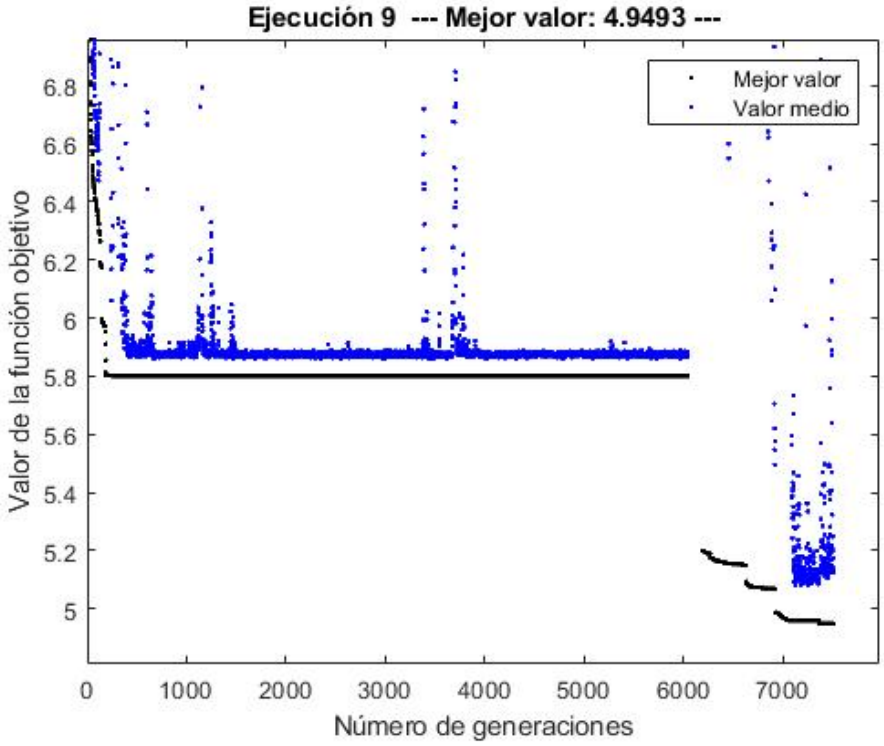


Figura 6.9 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la novena ejecución del algoritmo genético en el problema *Cassini1*.

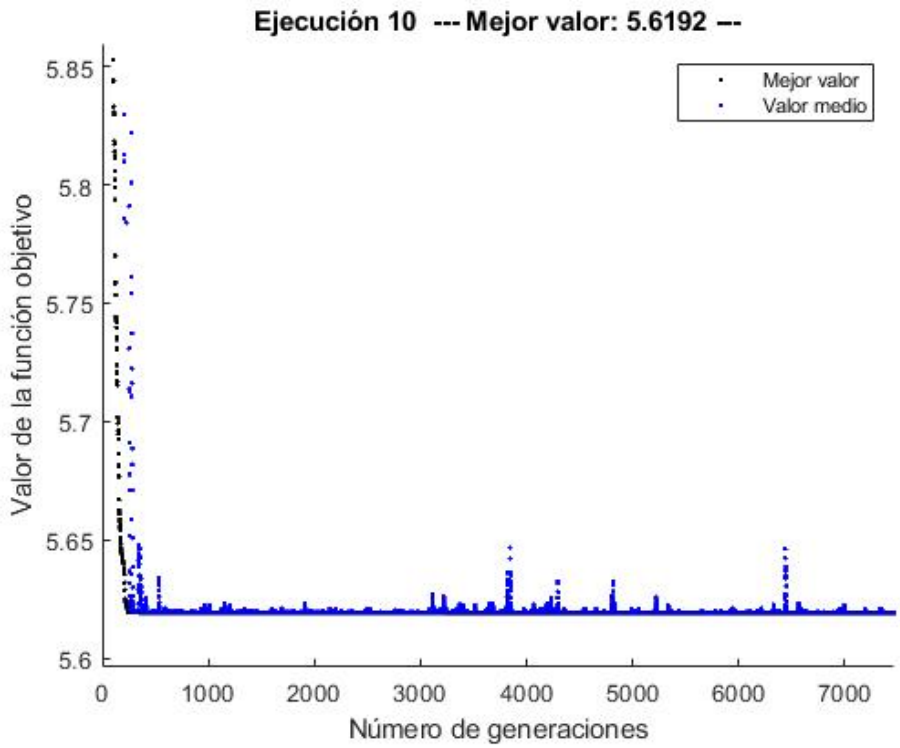


Figura 6.10 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la décima ejecución del algoritmo genético en el problema *Cassini1*.

Tabla 6.1 Resultados obtenidos durante las 10 ejecuciones juntos con los tiempos de ejecución.

EJECUCIONES	J (km/s)	Tiempo (min)
1	5.8622	41.2105
2	5.4071	38.4328
3	5.3111	38.8441
4	5.3121	38.6492
5	5.3132	38.6404
6	5.3129	38.8975
7	5.3130	38.8330
8	5.0595	38.9734
9	4.9493	39.2511
10	5.6192	40.3645

Las figuras demuestran que efectivamente las 7500 generaciones eran suficientes para la obtención de buenos resultados, ya que, a excepción de las ejecuciones 2 y 3, el algoritmo se había estancado antes de alcanzar el límite. Los tramos en los que el algoritmo genético está evolucionando hacia soluciones cada vez mejores se identifican rápidamente puesto que los puntos negros van descendiendo y además los valores de la media de cada generación varían unas décimas con respecto al mejor valor. Esa desviación de la media permite ver que el algoritmo ha explorado el espacio de búsqueda generando nuevos individuos que, por lo general, serán peores que el mejor de la generación de progenitores, pero también aparecerá alguno que mejore al mejor de los progenitores. En cambio, cuando el algoritmo se estanca, casi todos los individuos comienzan a ser muy parecidos entre sí, reduciéndose la posibilidad de evolución genética, y la media tiende a ser prácticamente

igual al mejor valor encontrado.

Respecto a los resultados, destacar que con un estudio de 6.5349 horas, se ha logrado alcanzar el resultado de 4.9493 km/s para la función objetivo que está lo suficientemente cerca del óptimo como para considerarlo un buen resultado. Véase en la Figura 6.9, donde se demuestra que gran parte de culpa del descubrimiento de esta solución la tiene la mutación y el cruce entre progenitores del algoritmo genético. Esa mutación aparece en las gráficas como pequeños saltos en la curva dibujada por los puntos negros.

Algoritmo de optimización por enjambre de partículas

Al igual que para el algoritmo genético van a recordarse los valores para los diferentes parámetros que definirán la actuación del algoritmo de optimización. Del Capítulo 5 se extraen:

- Factor de inercia: 1
- Coeficiente de atracción al mejor personal: 0.75
- Coeficiente de atracción al mejor global: 0.25
- Tamaño del enjambre: 1000 partículas

Si a esto se añade que como condición de parada se impone un número total de 4000 iteraciones (actualizaciones de posición), puede aplicarse sin más dilación el algoritmo para la optimización del problema. La razón de elegir 4000 iteraciones como límite vuelve a ser la comprobación empírica de que suele estancarse antes de llegar a dicho número.
A continuación se muestran las diez figuras de las ejecuciones realizadas y, como cierre, una tabla que recoge todos los resultados.

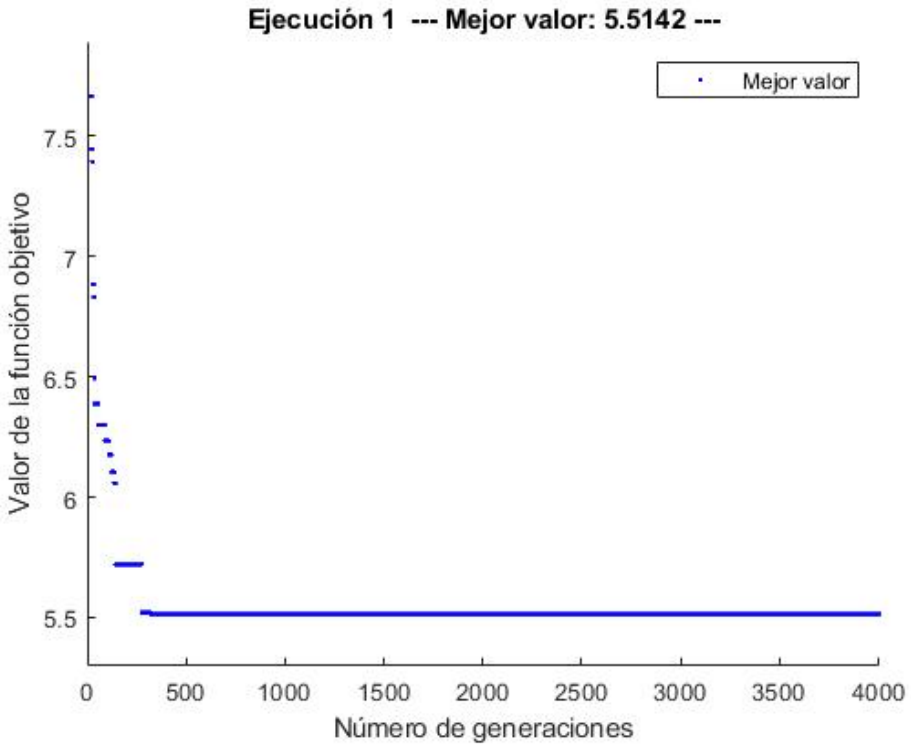


Figura 6.11 Representación de los mejores valores para la primera ejecución del algoritmo en el problema *Cassini1*.

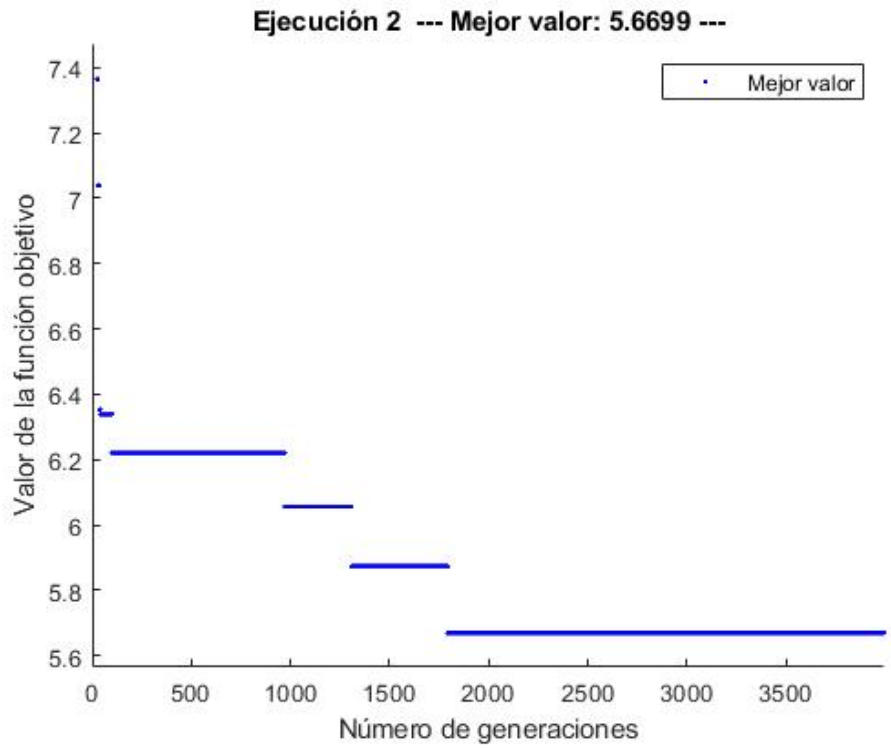


Figura 6.12 Representación de los mejores valores para la segunda ejecución del algoritmo en el problema *Cassini1*.

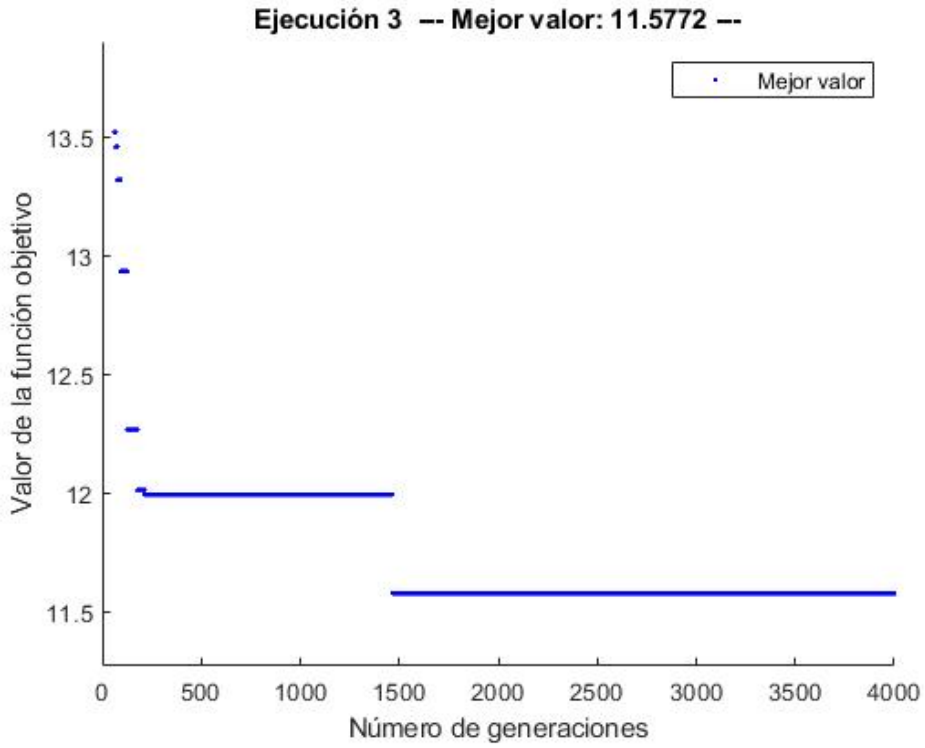


Figura 6.13 Representación de los mejores valores para la tercera ejecución del algoritmo en el problema *Cassini1*.

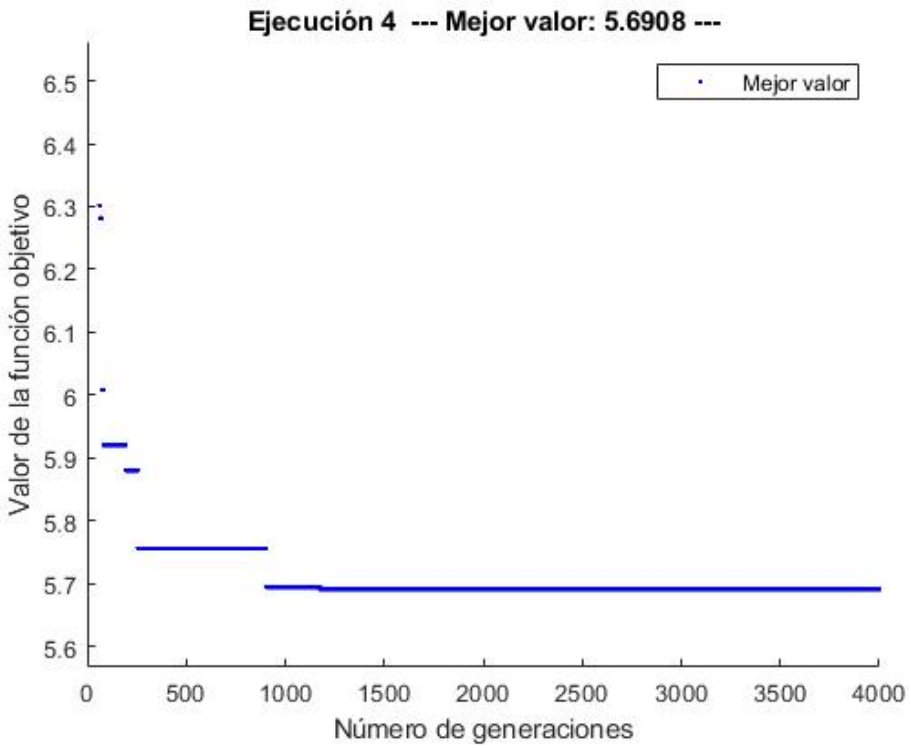


Figura 6.14 Representación de los mejores valores para la cuarta ejecución del algoritmo en el problema *Cassini1*.

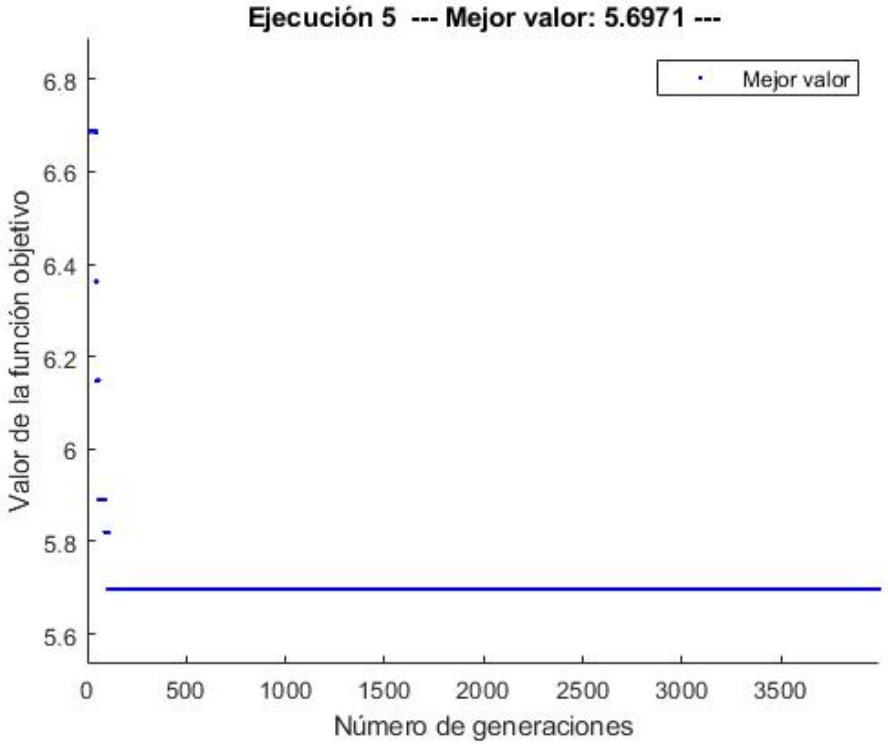


Figura 6.15 Representación de los mejores valores para la quinta ejecución del algoritmo en el problema *Cassini1*.

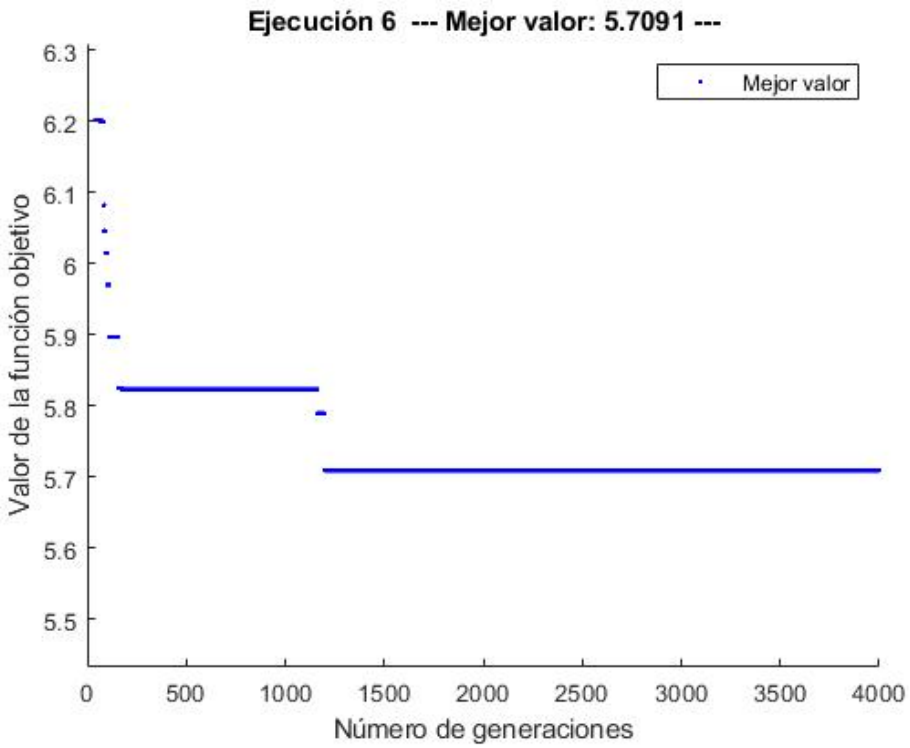


Figura 6.16 Representación de los mejores valores para la sexta ejecución del algoritmo en el problema *Cassini1*.

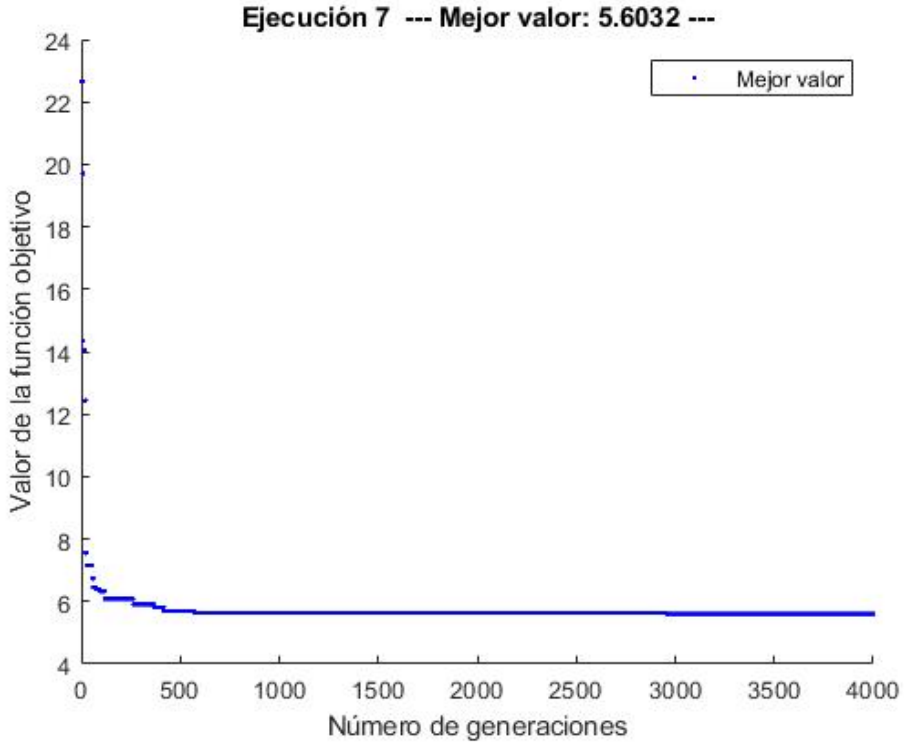


Figura 6.17 Representación de los mejores valores para la séptina ejecución del algoritmo en el problema *Cassini1*.

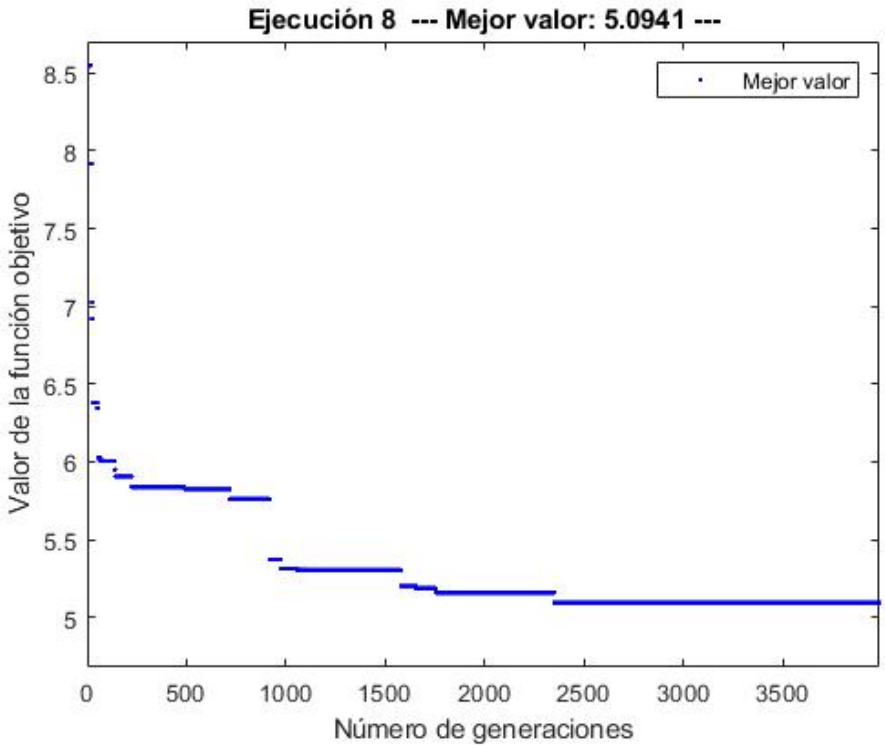


Figura 6.18 Representación de los mejores valores para la octava ejecución del algoritmo en el problema *Cassini1*.

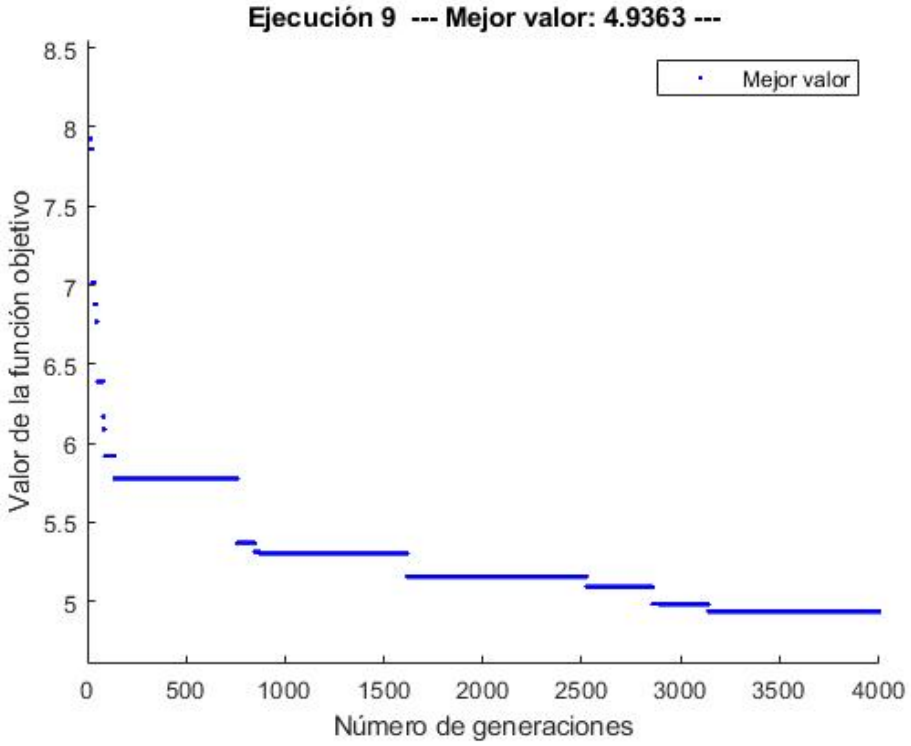


Figura 6.19 Representación de los mejores valores para la novena ejecución del algoritmo en el problema *Cassini1*.

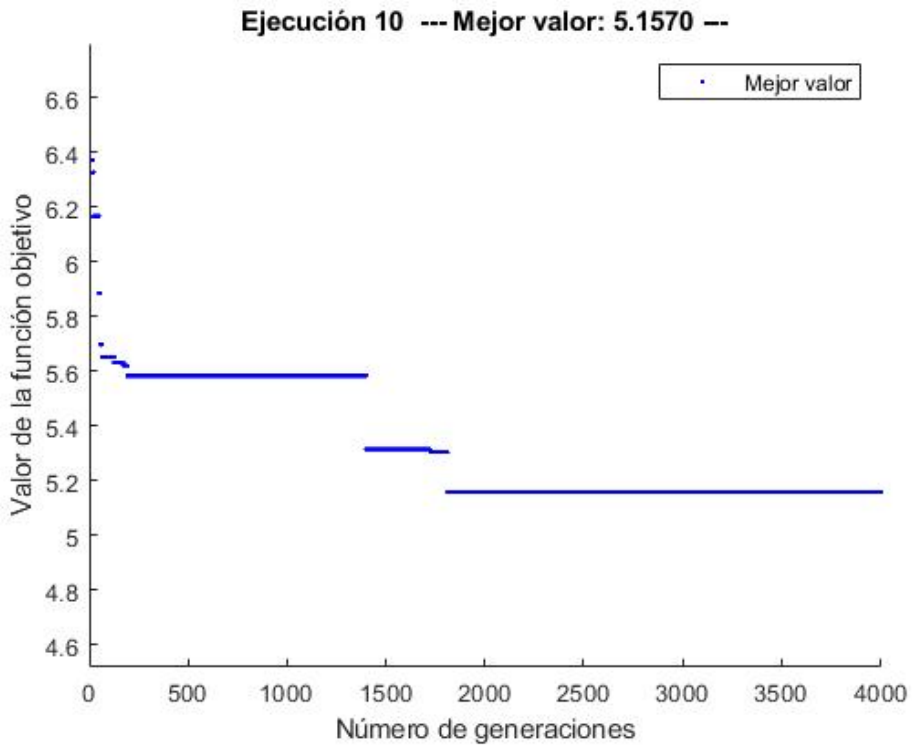


Figura 6.20 Representación de los mejores valores para la décima ejecución del algoritmo en el problema *Cassini1*.

Tabla 6.2 Resultados obtenidos durante las 10 ejecuciones juntos con los tiempos de ejecución.

EJECUCIONES	J (km/s)	Tiempo (min)
1	5.5142	24.4472
2	5.6699	23.2867
3	11.5772	23.1737
4	5.6908	23.4300
5	5.6971	23.1906
6	5.7091	23.2141
7	5.6031	23.2130
8	5.0941	25.0376
9	4.9363	24.5985
10	5.1570	24.8440

En el caso de este algoritmo, la función de MATLAB no da la opción de representar la media de todos los valores de la función objetivo que proporcionan las partículas en cada iteración, luego no es posible analizar, en principio, el nivel de exploración o explotación que tiene el algoritmo. Aunque observando las gráficas extraídas, puede verse cómo los mejores valores se van encontrando de manera que los puntos representados no forman una curva continua descendente si no que aparece una función escalonada. Esto nos permite deducir que el algoritmo tiende más a la exploración que a la explotación de zonas donde se han encontrado buenos resultados, ya que suele encontrar mejores resultados de manera aleatoria que mediante una mejora continua de los previamente encontrados. Gracias a los análisis realizados previamente (véase el Capítulo 5), el comportamiento con tendencia aleatoria de este algoritmo poco tiene que ver con el comportamiento propio de un

algoritmo aleatorio.

Independientemente de las características de búsqueda que tiene el algoritmo, los resultados obtenidos son realmente interesantes. Tras unas 3.6037 horas de cálculos, se consiguen representar las diez figuras junto con los valores de la Tabla 6.2, donde puede verse que el mejor de todos ellos proporciona un valor de 4.9363 km/s para la función objetivo.

Algoritmo mixto

Como ya se ha introducido, este algoritmo realiza una secuenciación de ejecuciones de algoritmo genético seguidas de ejecuciones del algoritmo de optimización por enjambre de partículas. Esta secuencia se repite hasta alcanzar un cierto número de ejecuciones de cada uno de los programas. El número de generaciones (GA) o de iteraciones (PSO) que van a realizarse en cada ejecución serán 300 y 200 respectivamente. Estos valores se han escogido ya que, en esos primeros cálculos de cada algoritmo, es cuando más mejora se produce durante las optimizaciones ya expuestas. Es cierto que a medida que avanza la optimización el algoritmo se va estancando como cualquier otro, pero era necesario escoger unas limitaciones bajas puesto que se realizarán 10 secuencias GA-PSO para completar 5000 iteraciones en total. Por tanto, para el análisis general se realizarán 10 ejecuciones (cada una de 5000 generaciones) como se hizo para los algoritmos anteriores.

Previamente a mostrar las figuras resultantes, aclarar que los parámetros utilizados en cada uno de los algoritmos independientes son los ya comentados en cada uno de sus apartados, puesto que son los que dan lugar al os mejores resultados.

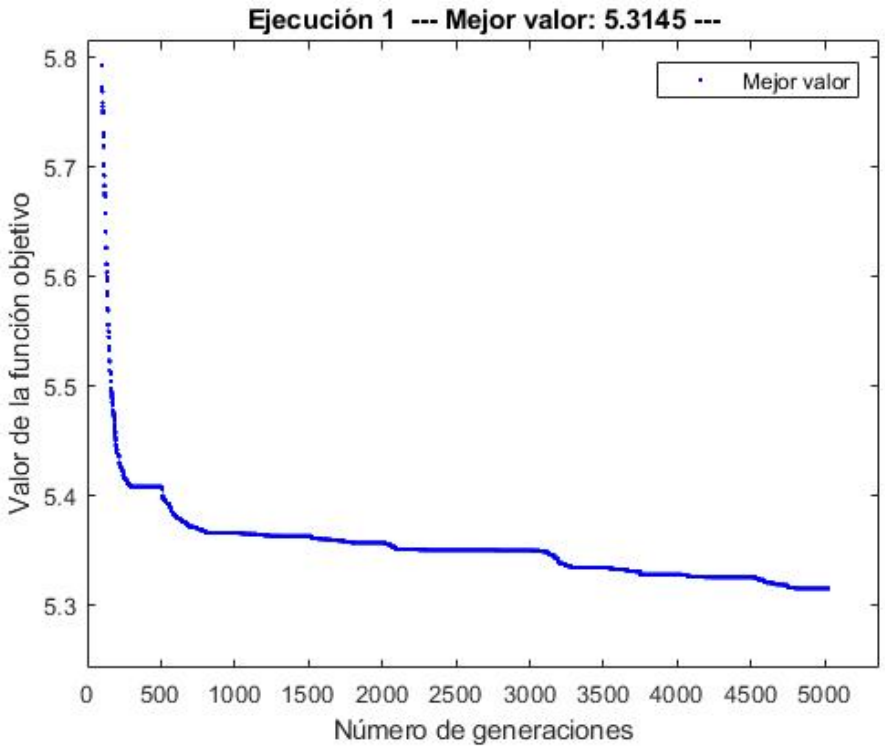


Figura 6.21 Representación de los mejores valores para la primera ejecución del algoritmo mixto en el problema *Cassini1*.

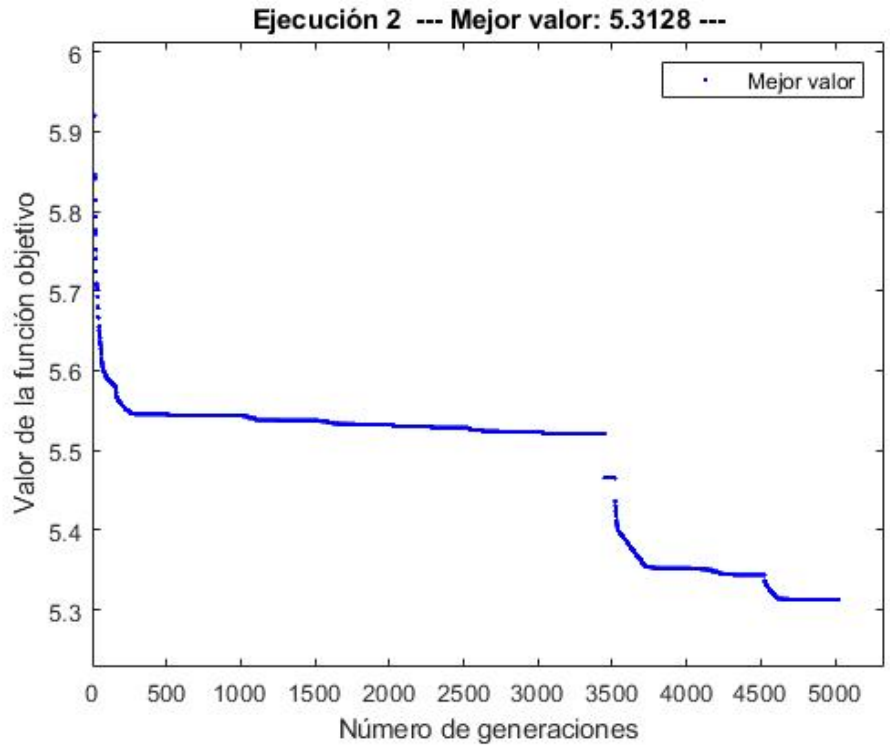


Figura 6.22 Representación de los mejores valores para la segunda ejecución del algoritmo mixto en el problema *Cassini1*.

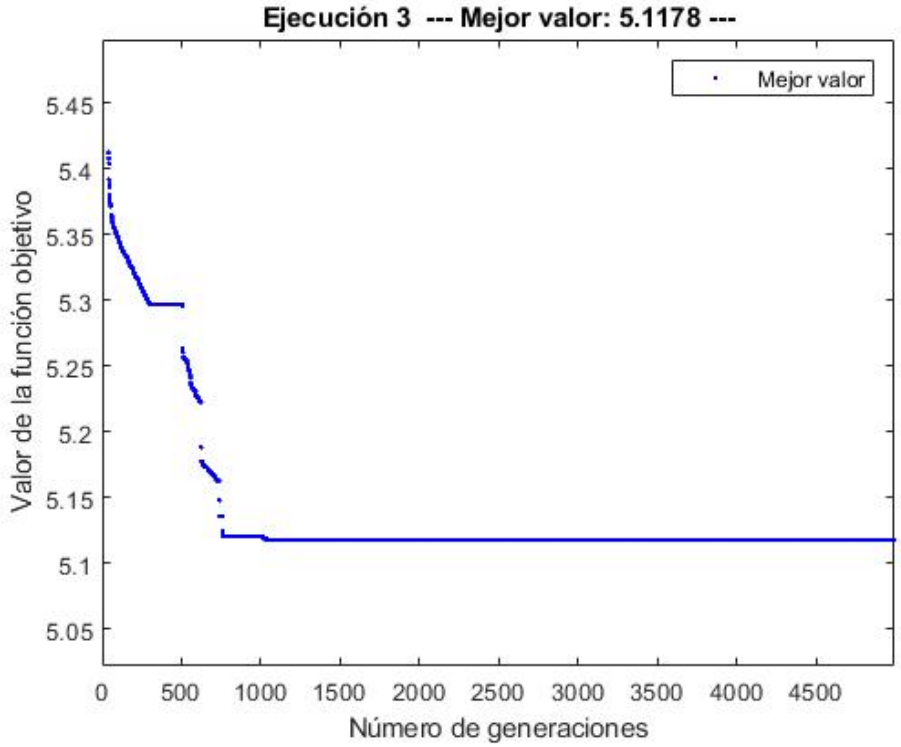


Figura 6.23 Representación de los mejores valores para la tercera ejecución del algoritmo mixto en el problema *Cassini1*.

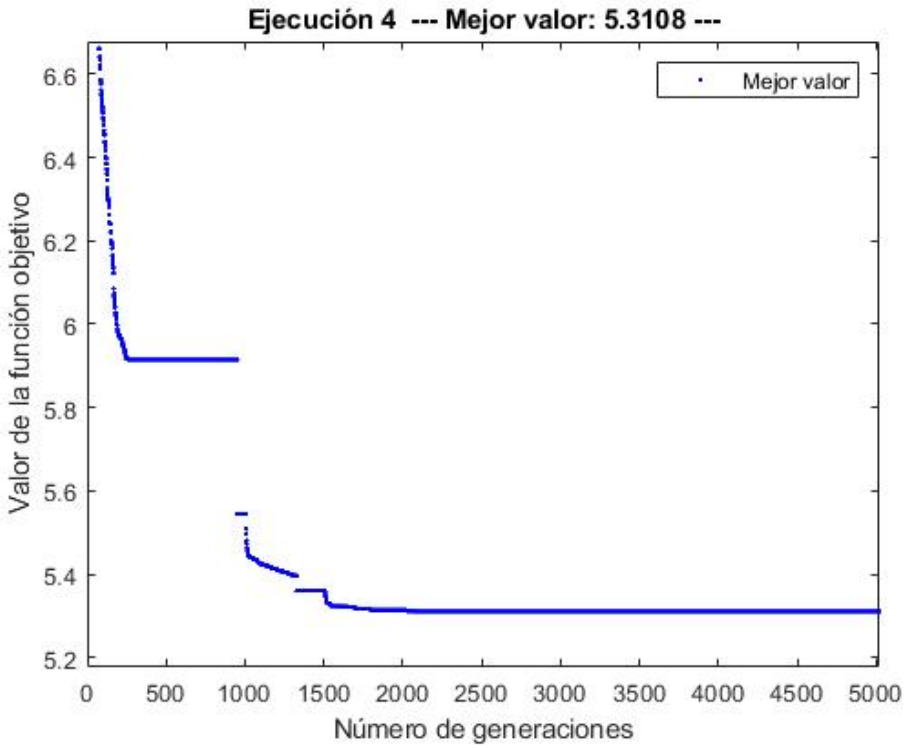


Figura 6.24 Representación de los mejores valores para la cuarta ejecución del algoritmo mixto en el problema *Cassini1*.

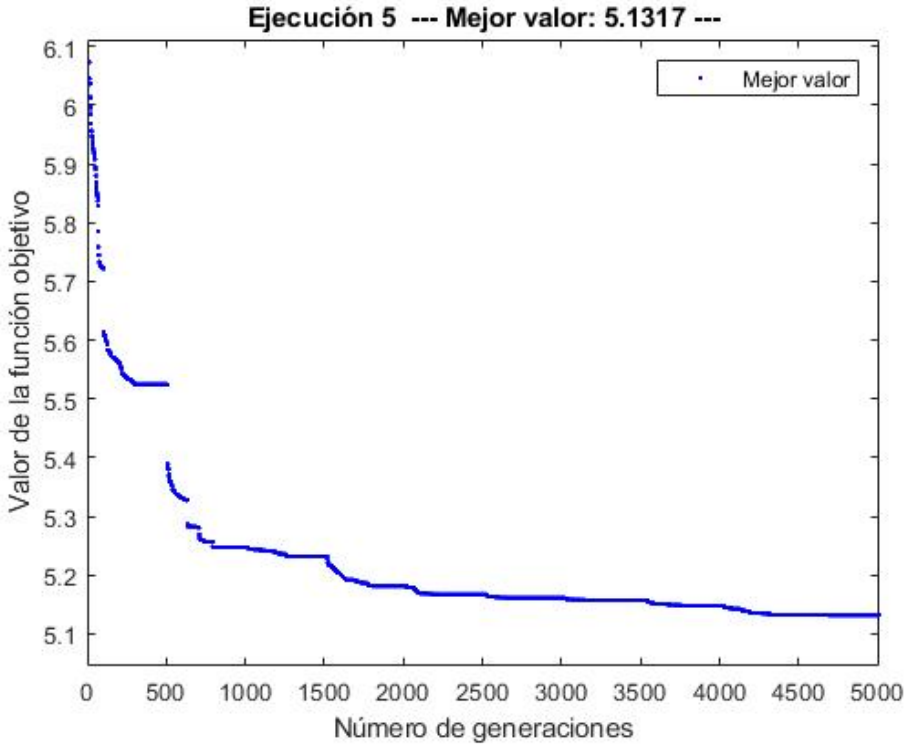


Figura 6.25 Representación de los mejores valores para la quinta ejecución del algoritmo mixto en el problema *Cassini1*.

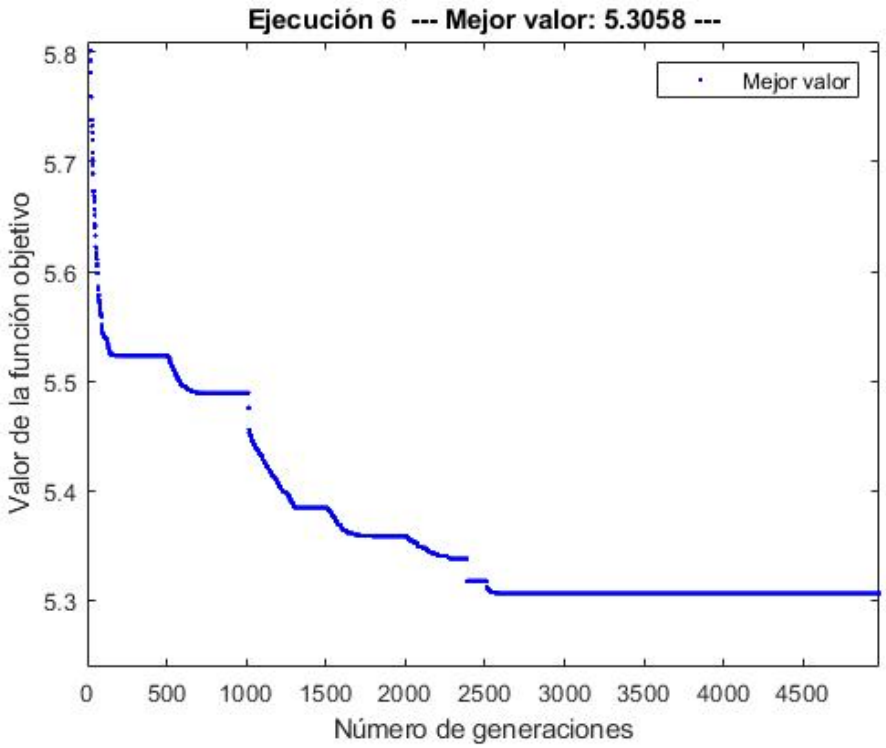


Figura 6.26 Representación de los mejores valores para la sexta ejecución del algoritmo mixto en el problema *Cassini1*.

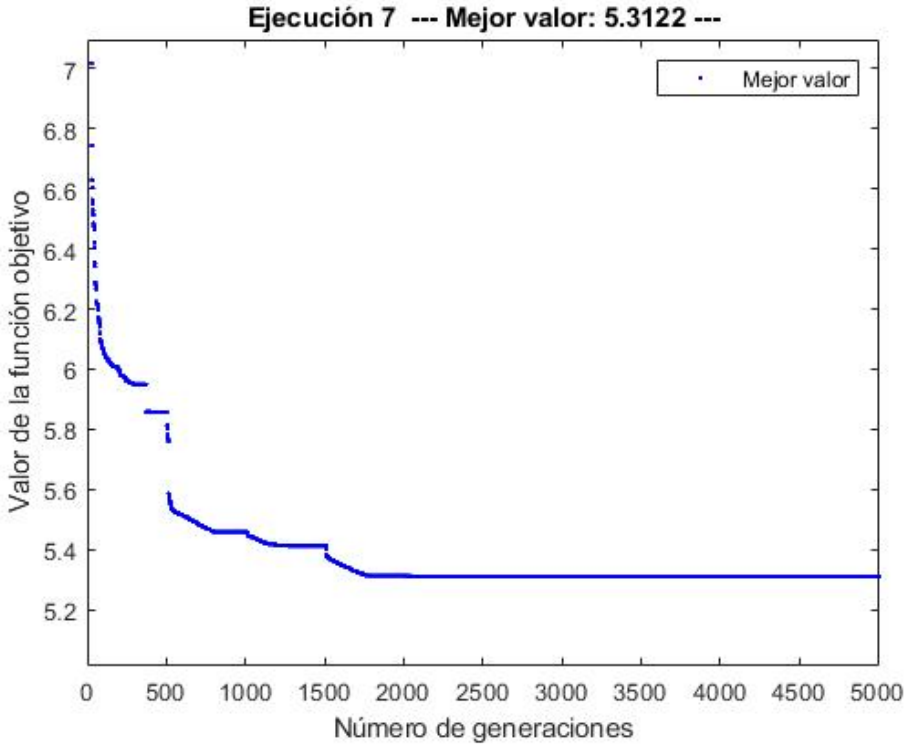


Figura 6.27 Representación de los mejores valores para la séptima ejecución del algoritmo mixto en el problema *Cassini1*.

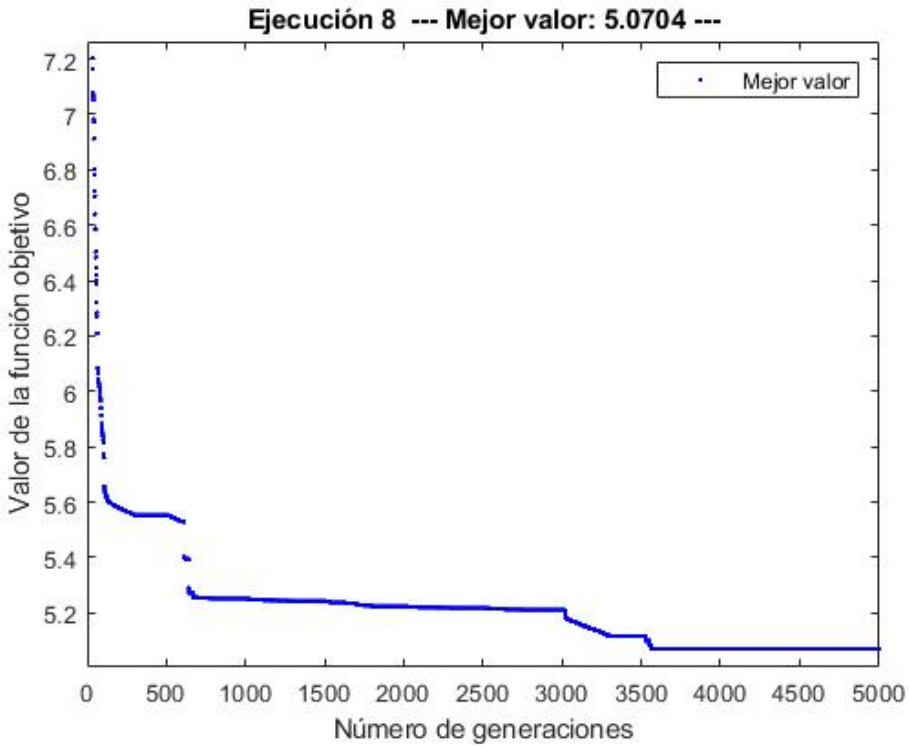


Figura 6.28 Representación de los mejores valores para la octava ejecución del algoritmo mixto en el problema *Cassini1*.

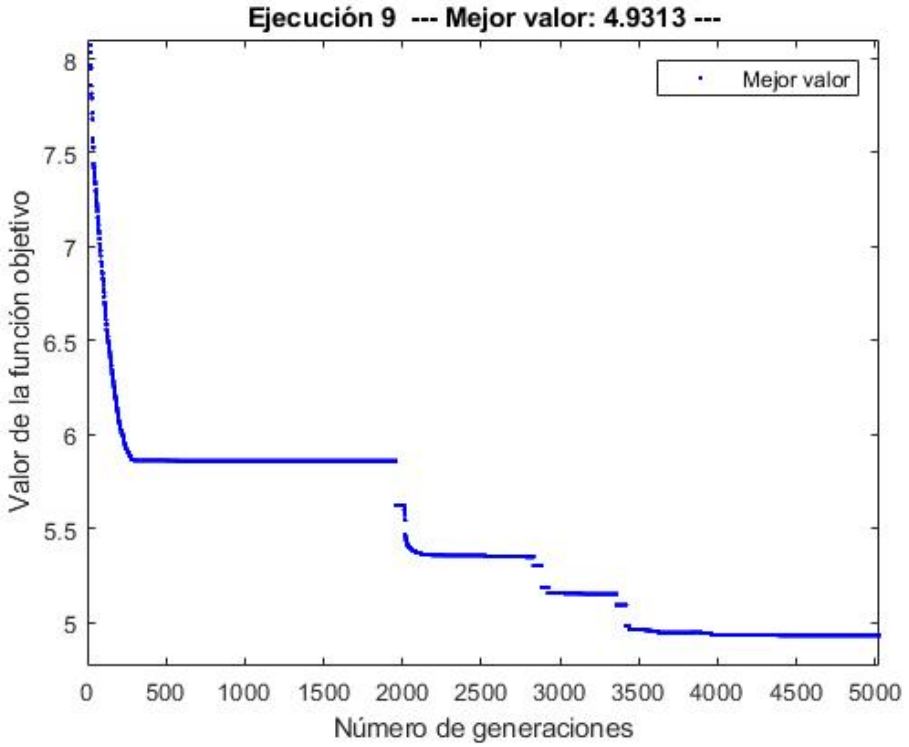


Figura 6.29 Representación de los mejores valores para la novena ejecución del algoritmo mixto en el problema *Cassini1*.

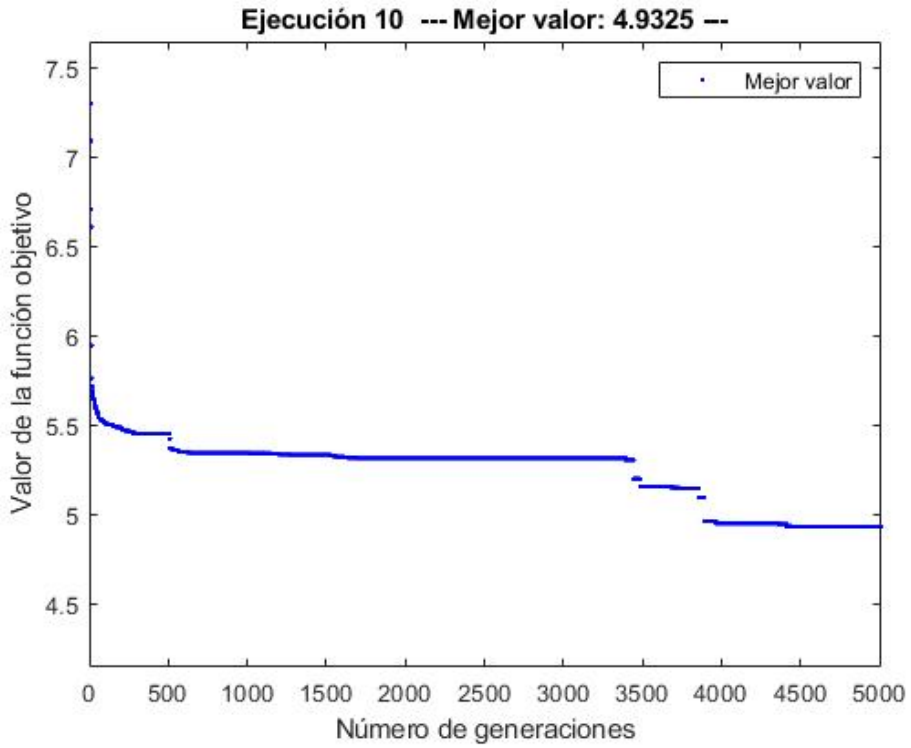


Tabla 6.3 Resultados obtenidos durante las 10 ejecuciones juntos con los tiempos de ejecución.

EJECUCIONES	J (km/s)	Tiempo (min)
1	5.3145	28.9695
2	5.3128	27.9732
3	5.1178	27.8375
4	5.3108	27.8585
5	5.1317	27.8205
6	5.3058	27.6295
7	5.3122	27.7478
8	5.0704	27.6799
9	4.9313	27.0073
10	4.9325	27.0486

Según se ha podido ver en las figuras presentadas, el algoritmo funciona correctamente ya que se evoluciona hacia resultados mejores con el paso de las generaciones. En dicho proceso de búsqueda se va combinando la evolución genética que proporciona el GA con la búsqueda por el movimiento de las partículas del enjambre del PSO. Las discontinuidades observadas suelen estar provocadas por la búsqueda mediante partículas aunque también es posible que el GA genere discontinuidades debido a la mutación.

La aplicación de este algoritmo novedoso ha provocado que se encuentre una solución mejor que las encontradas por los algoritmos por separado. Dicha solución, 4.9313 km/s para la función objetivo, se ha alcanzado tras unas 4.6262 horas de optimización mediante las ejecuciones realizadas.

Comparativa y lógica de resultados

En este último apartado de resultados para el problema *Cassini1*, van a compararse los resultados obtenidos mediante los tres algoritmos utilizados con los resultados publicados en [4]. Además, se muestran las componentes de los vectores de decisión para cada uno de los casos y se estudian los tiempos totales que duraría la misión.

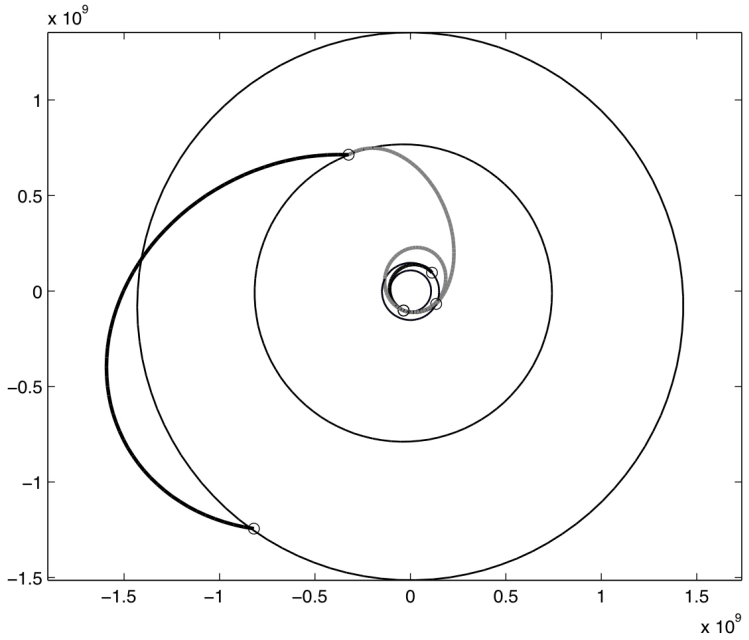


Figura 6.30 Trayectoria seguida para la mejor solución encontrada del problema Cassini1. Extraída de [15]. Órbitas representadas: Venus, Tierra, Júpiter, Saturno.

Tabla 6.4 Mejores resultados obtenidos en el trabajo y publicados en la página de la ESA [4].

RESULTADOS DEL TRABAJO		RESULTADOS PUBLICADOS	
ALGORITMO	MEJOR VALOR	ALGORITMO	MEJOR VALOR
GA	4.9493 km/s	MBH	5.0088 km/s
PSO	4.9363 km/s	DE	4.9340 km/s
MIXTO	4.9313 km/s	PSO	4.9307 km/s
ALEATORIO	8.3269 km/s		

Comenzando por la comparación entre resultados, se recogen en la Tabla 6.4 tanto los obtenidos en este trabajo como los obtenidos por otros investigadores que dedicaron su tiempo a encontrar el valor óptimo para las funciones objetivo de los problemas propuestos. Con una visión general puede comprobarse que se han mejorado dos de los resultados publicados oficialmente utilizando algoritmos basados en tiempo de ejecución pequeños con pocas generaciones. Es probable que para alcanzar el óptimo del problema sea necesario un análisis de parámetros de cada algoritmo mucho más exhaustivo o bien establecer un límite de generaciones mucho mayor en los algoritmos utilizados lo que conllevaría a ejecuciones mucho más duraderas. Como el objetivo principal del trabajo es obtener un resultado cercano al óptimo en el menor tiempo posible, puede concluirse que se ha conseguido lograr. La razón por la que se desechó intentar igualar los resultados óptimos publicados radica en la falta de medios con la suficiente potencia de cálculo como para realizar

ejecuciones de los algoritmos mucho más duraderas o con poblaciones y enjambres mucho mayores, que favorecerían el hallazgo de dicho punto óptimo. Si se comparan los resultados obtenidos en este trabajo se observa como, para este primer problema, el algoritmo que mejor ha funcionado ha sido el mixto, luego parece que ha sido una buena idea su introducción. Además, un claro ejemplo de que los tres algoritmos han funcionado bien se tiene al comparar con el resultado del algoritmo aleatorio. Dicho algoritmo aleatorio se ha ejecutado de forma que calcule los mismos puntos que el resto, pero aún así genera un valor mucho peor.

En cuanto a los vectores de decisión logrados, que evaluándolos en la función objetivo dan como resultado los expuestos en la Tabla 6.4, se han recogido por componentes en la Tabla 6.5.

Tabla 6.5 Componentes de los vectores de decisión que dan lugar a los mejores resultados de la función objetivo hallados.

RESULTADO	COMPONENTES DEL VECTOR DE DECISIÓN					
J (x) (km/s)	t_0 (MJD2000)	T_1 (Días)	T_2 (Días)	T_3 (Días)	T_4 (Días)	T_5 (Días)
4.9493	-788.764967	163.782977	449.385759	55.566917	976.308125	4459.711526
4.9363	-787.514043	156.100230	449.385836	54.685496	1025.007727	4553.282742
4.9313	-789.007680	157.069817	449.385871	54.685374	1025.009061	4553.284923

Conocidos los tiempos de vuelos interplanetarios que optimizan la misión Cassini, es posible calcular cuánto tardaría la sonda en alcanzar el planeta Saturno en el caso de realizarse la misión cumpliendo con las restricciones impuestas en su modelado y con los tiempos de vuelos obtenidos. Para calcular el tiempo de la misión, basta con sumar los tiempos de vuelos interplanetarios (T_1, ..., T_5).

Tabla 6.6 Duración de la misión en función del valor de la función objetivo deseado.

RESULTADO	DURACIÓN DE LA MISIÓN
J (x) (km/s)	T_{total} (Años)
4.9493	16.7254
4.9363	17.0917
4.9313	17.0943

Como se esperaba, viendo los tiempos de la Tabla 6.6, conforme mayor es el total de impulsos (mayor combustible gastado) menor será el tiempo empleado en alcanzar el destino. El ahorro de combustible conllevará a realizar trayectorias que se sirvan de los impulsos gratuitos que proporcionan los planetas cuando se realiza una maniobra asistida por gravedad. Esto a su vez resulta en un aumento del tiempo que dura la misión.

La misión real duró unos 5 años y 9 meses, luego es posible deducir que no se realizó con el objetivo de minimizar los impulsos realizados o, lo que es lo mismo, minimizar el consumo de combustible. Lo que se hizo fue utilizar impulsos durante el trayecto para reducir el tiempo empleado en llegar a Saturno, ya que los trabajos de investigación que iba a llevar acabo la sonda una vez estuviera orbitando el planeta tendrían una duración prolongada. Es un ejemplo de que en la realidad no se busca la optimización según un parámetro concreto, si no que hay que llegar a un cierto equilibrio en el que todas las variables, que afectan fuertemente a los problemas, tengan valores que se adecúen a lo que se pretende desarrollar.

6.1.2 GTOC1

Este problema, descrito en el Capítulo 4, trata de maximizar el cambio de semieje mayor de la órbita del asteroide TW229. Como se ha expuesto en el Capítulo 4, dicho cambio está cuantificado por la función objetivo. El vector decisión en este problema es de 8 componentes, dos más que para el problema *Cassini1* ya que realizará dos sobrevuelos más. La principal diferencia con el problema anterior es que en este caso se busca maximizar la función objetivo, teniendo en cuenta que los algoritmos utilizados están programados para minimizar por defecto habrá que cambiar de signo a toda la función objetivo para que el mínimo teórico encontrado sea el máximo real cambiado de signo. Por ello, en las gráficas que se añaden el mejor valor está dado con signo negativo.

Para la presentación de resultados se sigue la misma estructura que para el problema anterior y se divide el apartado en tres secciones, una para cada algoritmo, junto con otra adicional para la comparación de los resultados obtenidos. Debido a las similitudes con el problema anterior, misiones sin maniobras en espacio profundo y vectores de decisión parecidos, se han utilizado los mismos parámetros exactamente para cada algoritmo, por lo que se comprueba si el análisis realizado de estos parámetros es válido para otros problemas.

Al utilizar los mismos algoritmos, el objetivo principal sigue siendo la obtención de resultados cercanos al óptimo publicado en un tiempo relativamente pequeño. Para la comprobación del funcionamiento de cada algoritmo para este problema, se han realizado de nuevo 10 ejecuciones de cada uno, cuyos resultados se exponen en las secciones que siguen.

Algoritmo genético

Antes de mostrar los resultados se recuerda que la limitación para el número de generaciones máximas se mantiene en 7500 siguiendo el mismo razonamiento que para el problema *Cassini1* y los parámetros que definirán la actuación del algoritmo genético son los calculados en el Capítulo 5 para los problemas sin maniobras en espacio profundo:

- Factor de mutación: 0.02
- Parámetro de cruce: 0.75
- Tamaño de la población: 1000 individuos
- Población de la élite: 1 % del total de la población

A continuación se muestran las figuras obtenidas para cada ejecución y una tabla resumen que recoge todos los resultados.

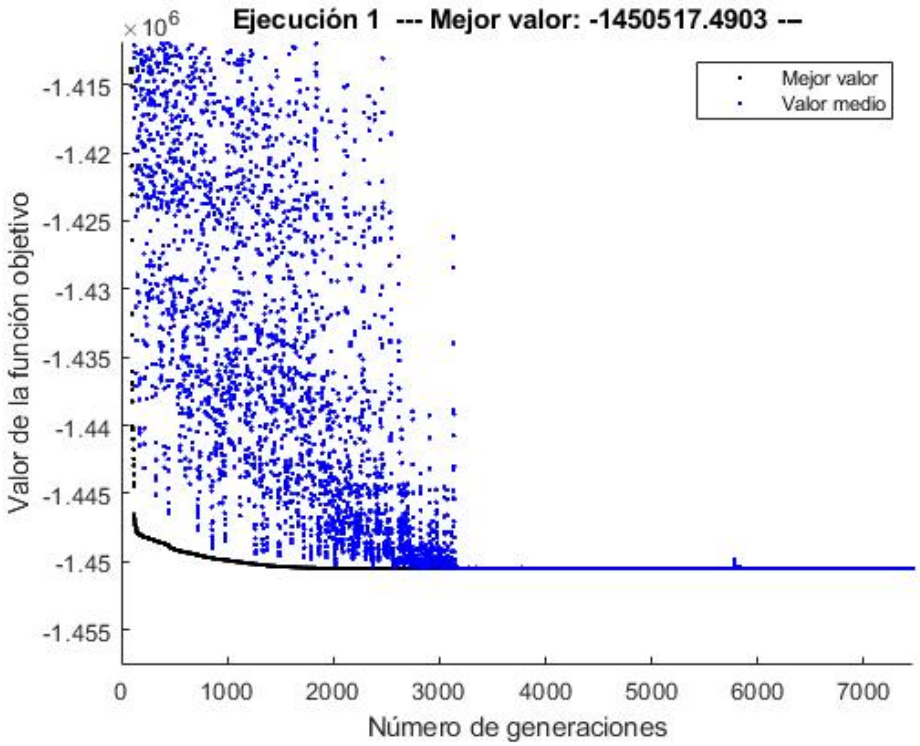


Figura 6.31 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la primera ejecución del algoritmo genético en el problema *GTOC1*.

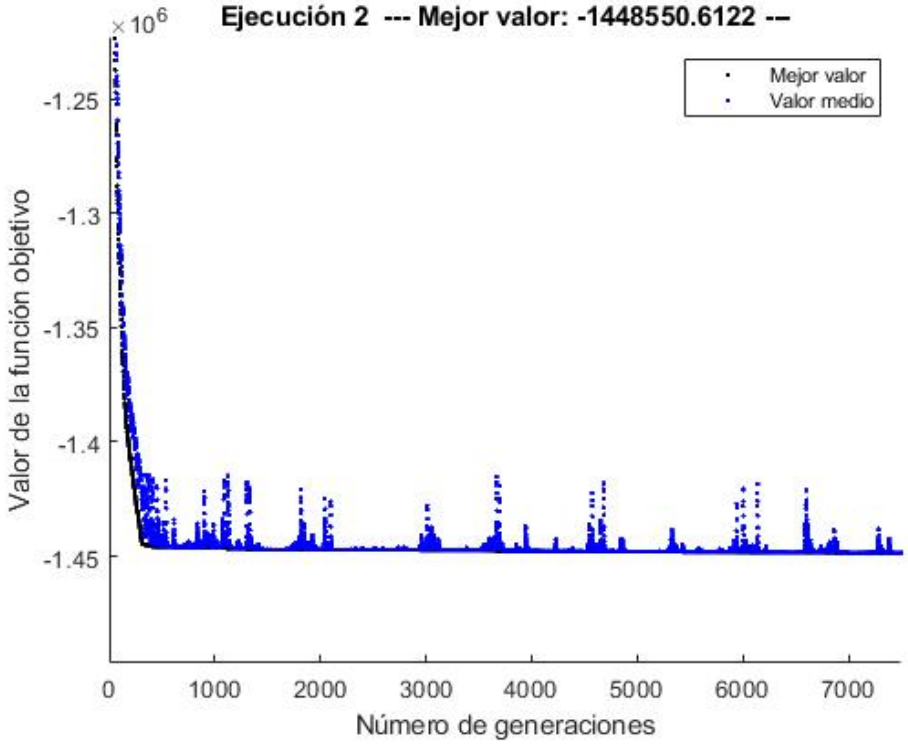


Figura 6.32 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la segunda ejecución del algoritmo genético en el problema *GTOC1*.

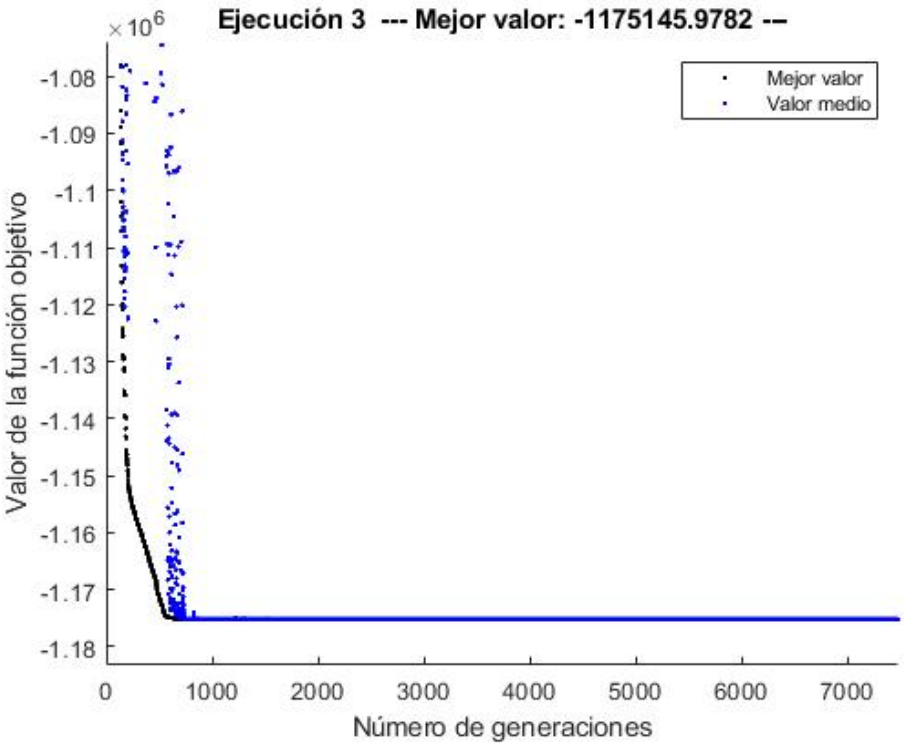


Figura 6.33 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la tercera ejecución del algoritmo genético en el problema *GTOC1*.

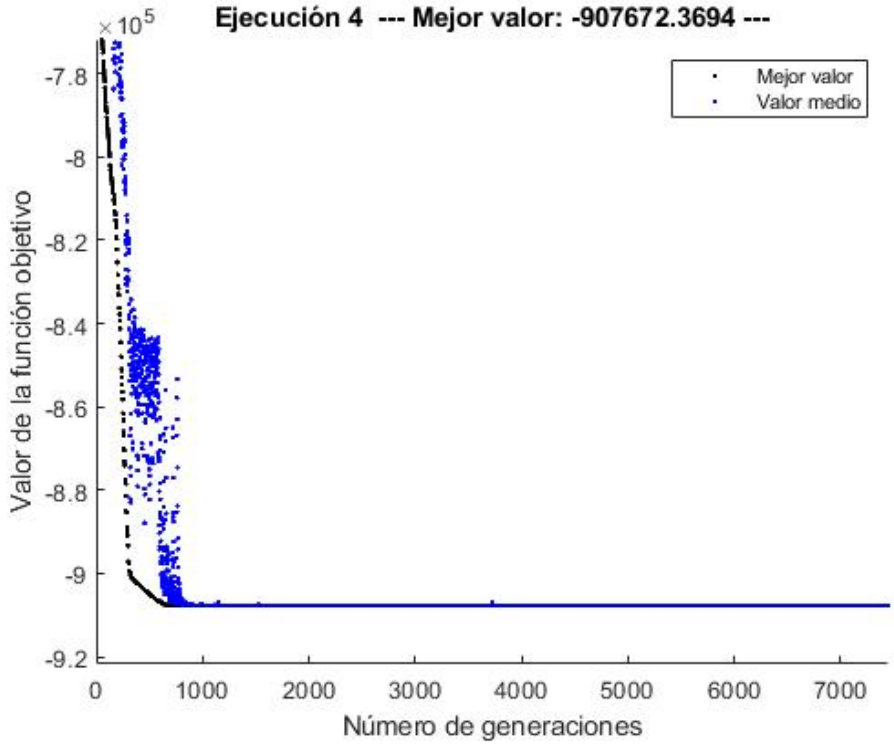


Figura 6.34 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la cuarta ejecución del algoritmo genético en el problema *GTOC1*.

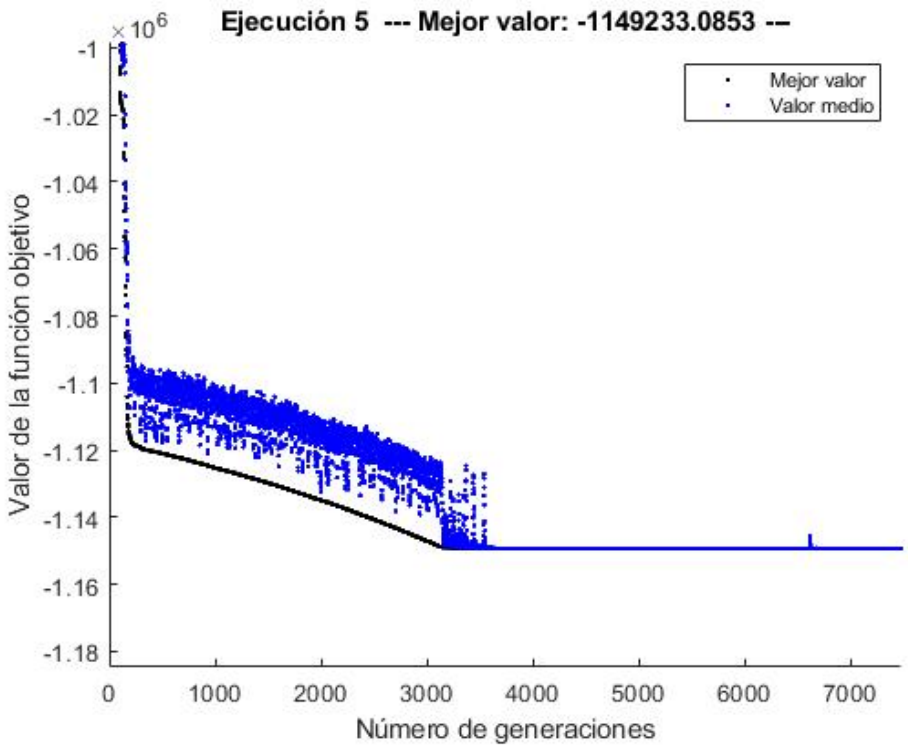


Figura 6.35 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la quinta ejecución del algoritmo genético en el problema *GTOC1*.

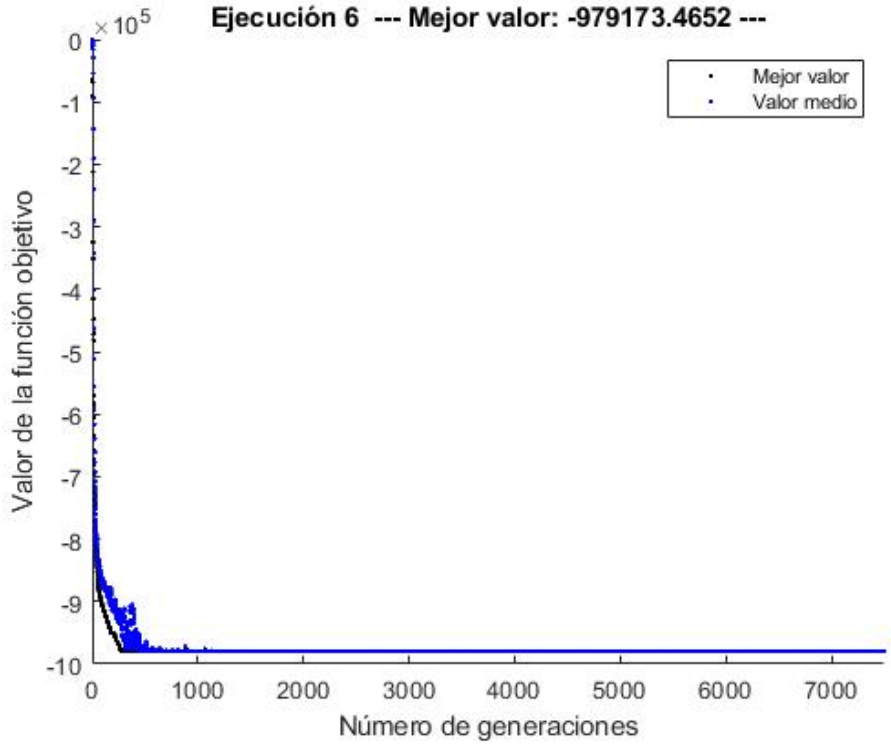


Figura 6.36 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la sexta del algoritmo genético en el problema *GTOC1*.

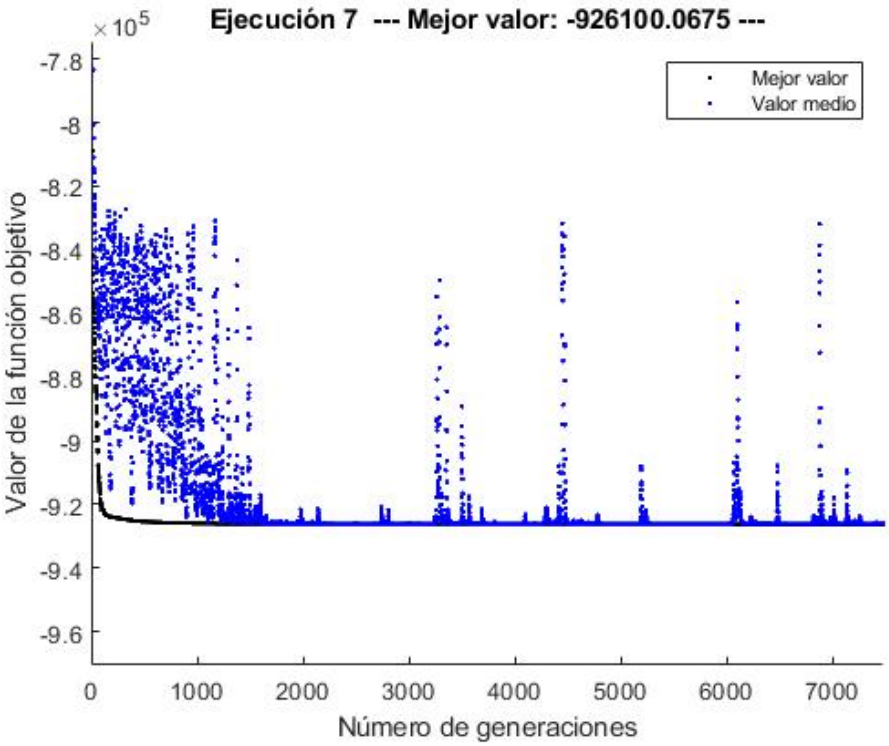


Figura 6.37 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la séptima ejecución del algoritmo genético en el problema *GTOC1*.

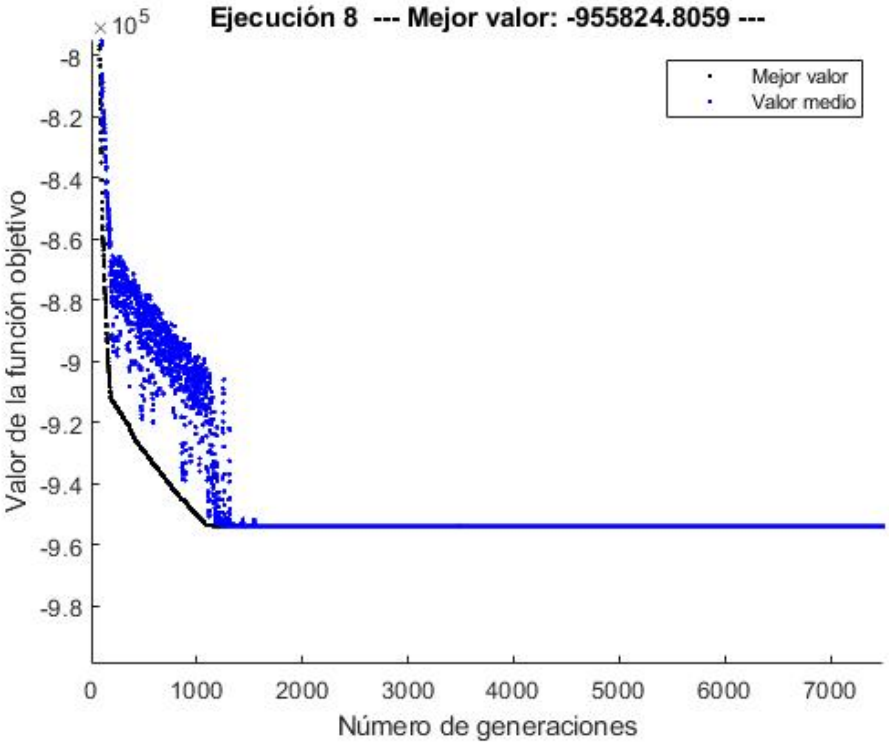


Figura 6.38 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la octava ejecución del algoritmo genético en el problema *GTOC1*.

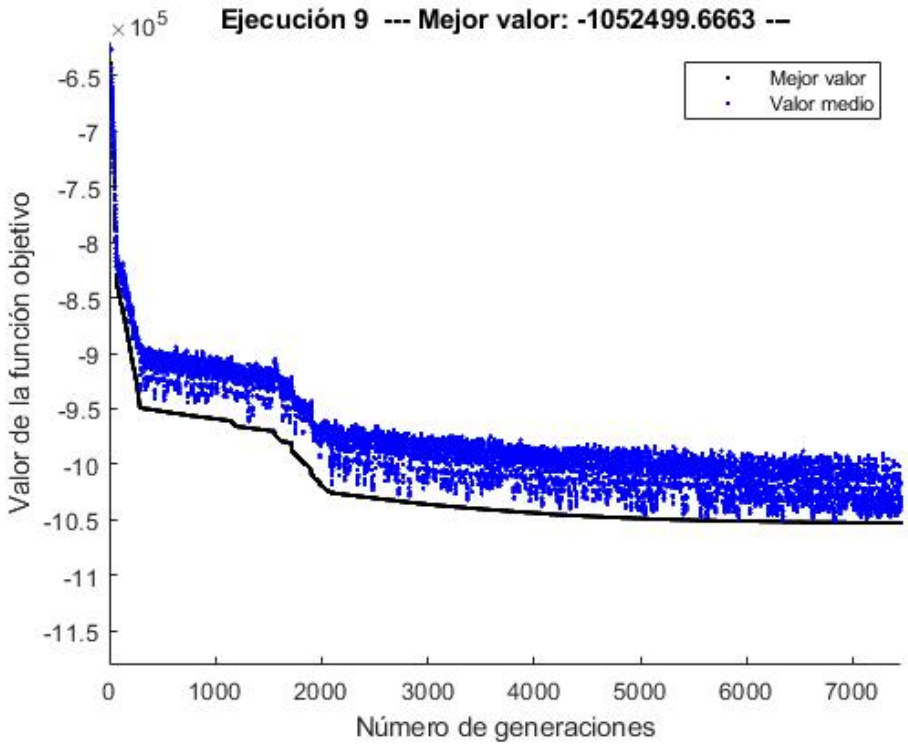


Figura 6.39 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la novena ejecución del algoritmo genético en el problema *GTOC1*.

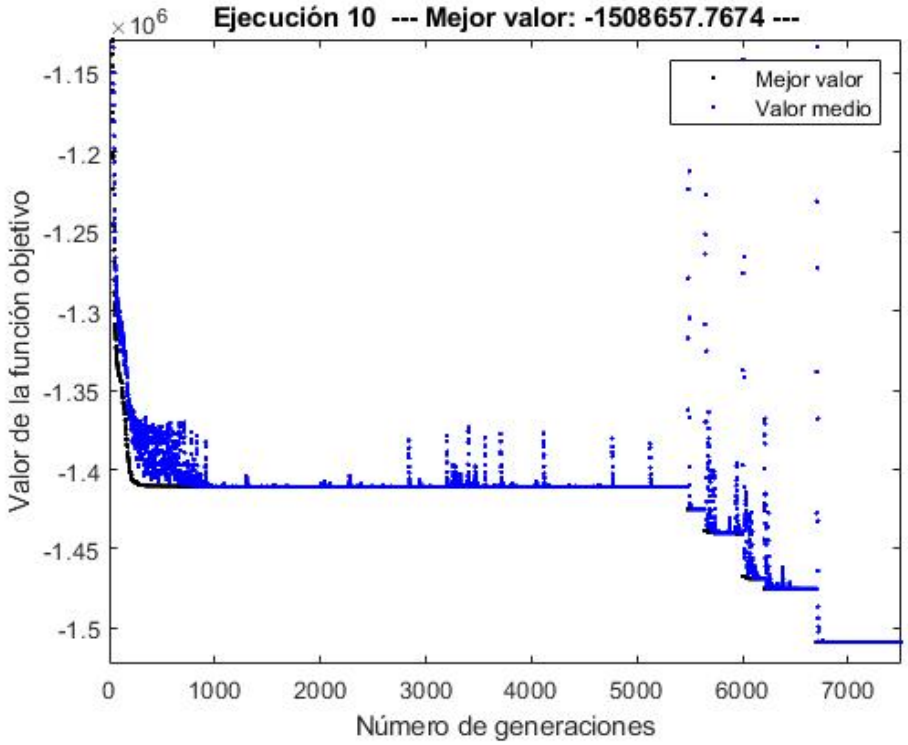


Figura 6.40 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la décima ejecución del algoritmo genético en el problema *GTOC1*.

Tabla 6.7 Resultados obtenidos durante las 10 ejecuciones junto con los tiempos de ejecución.

EJECUCIONES	J (julios)	Tiempo (min)
1	-1450517.4903	51.8860
2	-1448550.6122	52.0961
3	-1175145.9782	52.9908
4	-907672.3694	51.7726
5	-1149233.0853	52.6900
6	-979173.4652	52.2101
7	-926100.0676	52.3232
8	-953824.8059	52.3175
9	-1052499.6663	52.6199
10	-1508657.7674	50.8257

Observése que las diferencias entre los puntos de la media (puntos azules) y los mejores valores (puntos negro) que se aprecian en cada ejecución son de órdenes parecidos, pero en la representación hay ocasiones en las que parece mayor porque cambia la escala del Eje *y*.

Por último, de los resultados obtenidos, es destacable el valor de 1508657.7674 julios (con su signo adecuado) de la última ejecución que vuelve a encontrarse debido a la mutación en el algoritmo genético puesto que se observa un escalón en la curva que representan los puntos de mejor valor encontrado. Este valor es el máximo encontrado por el algoritmo genético en un tiempo de cálculo de unas 8.6955 horas.

Algoritmo de optimización por enjambre de partículas

Análogamente al algoritmo genético, en este algoritmo también se mantienen los parámetros usados para el problema *Cassini1*, un número máximo de iteraciones de 4000 y los parámetros del Capítulo 5:

- Factor de inercia: 1
- Coeficiente de atracción al mejor personal: 0.75
- Coeficiente de atracción al mejor global: 0.25
- Tamaño del enjambre: 1000 partículas

Los resultados se exponen a continuación.

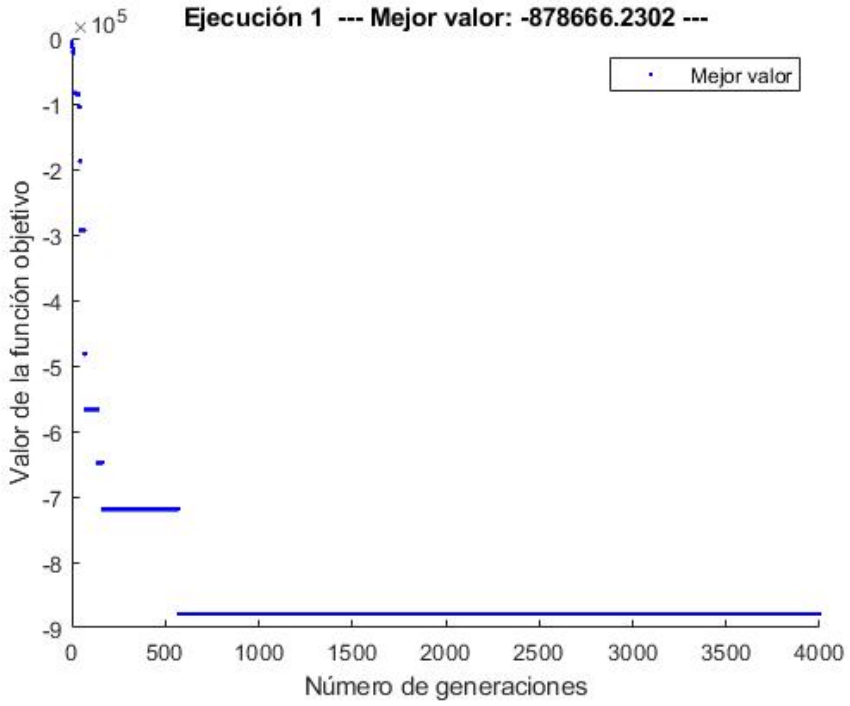


Figura 6.41 Representación de los mejores valores para la primera ejecución del algoritmo en el problema *GTOC1*.

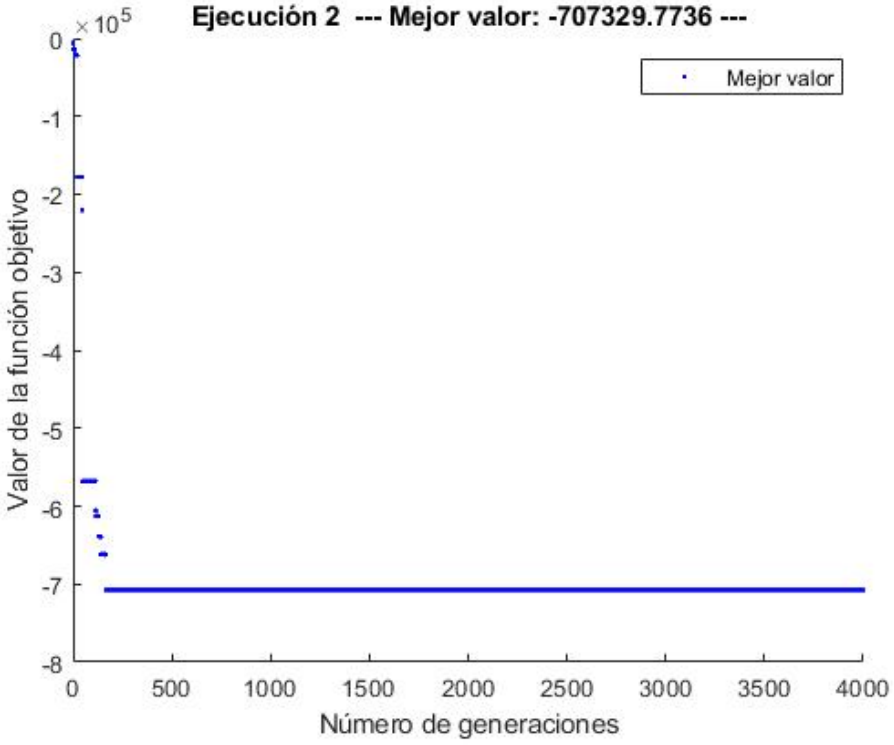


Figura 6.42 Representación de los mejores valores para la segunda ejecución del algoritmo en el problema *GTOC1*.

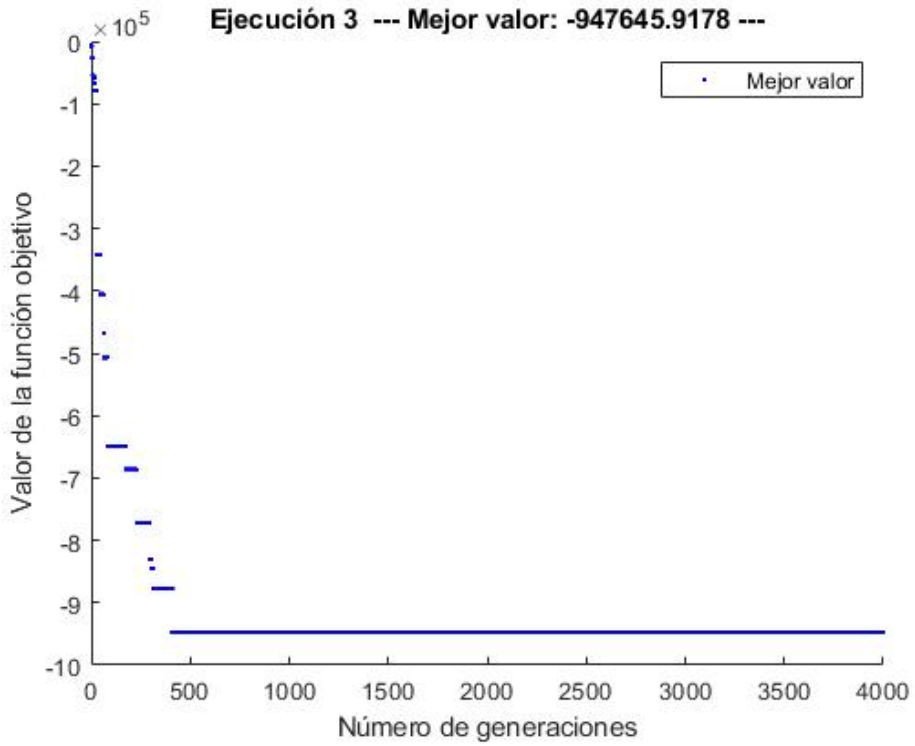


Figura 6.43 Representación de los mejores valores para la tercera ejecución del algoritmo en el problema *GTOC1*.

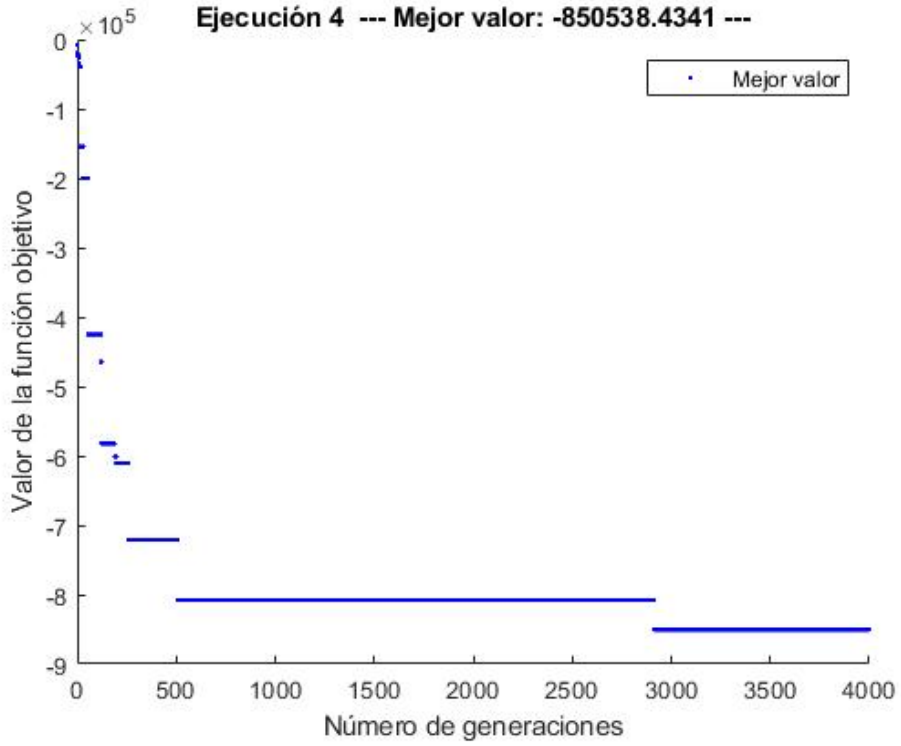


Figura 6.44 Representación de los mejores valores para la cuarta ejecución del algoritmo en el problema *GTOC1*.

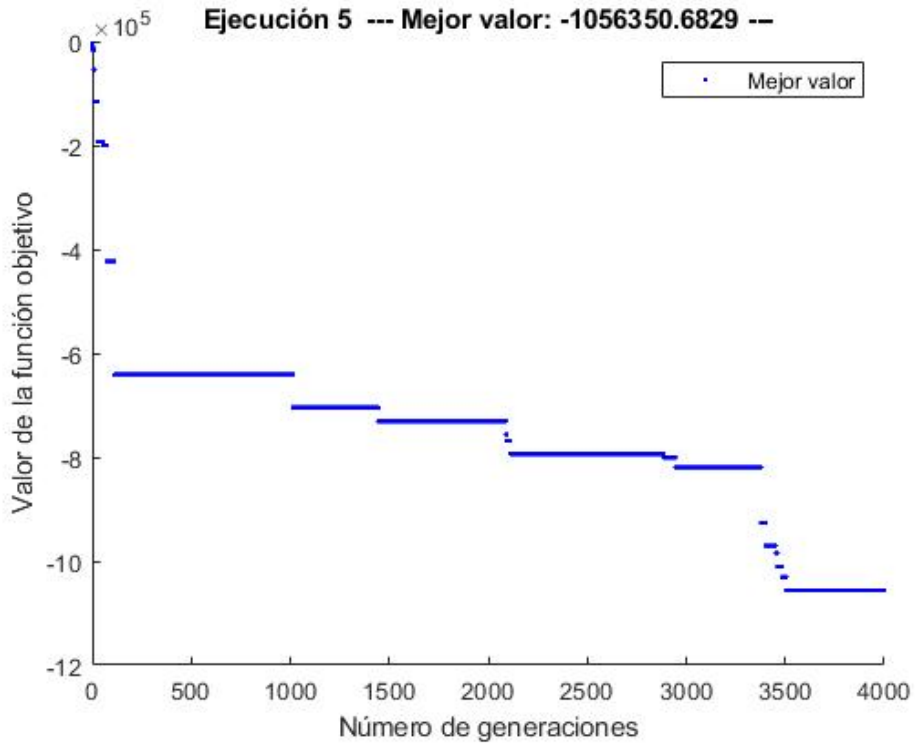


Figura 6.45 Representación de los mejores valores para la quinta ejecución del algoritmo en el problema *GTOC1*.

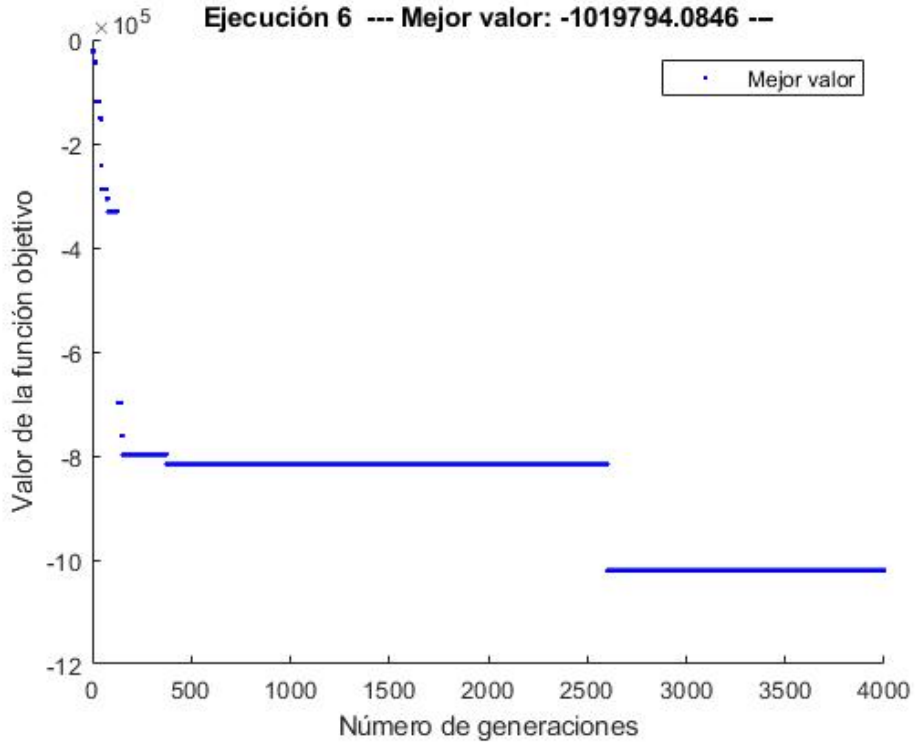


Figura 6.46 Representación de los mejores valores para la sexta ejecución del algoritmo en el problema *GTOC1*.

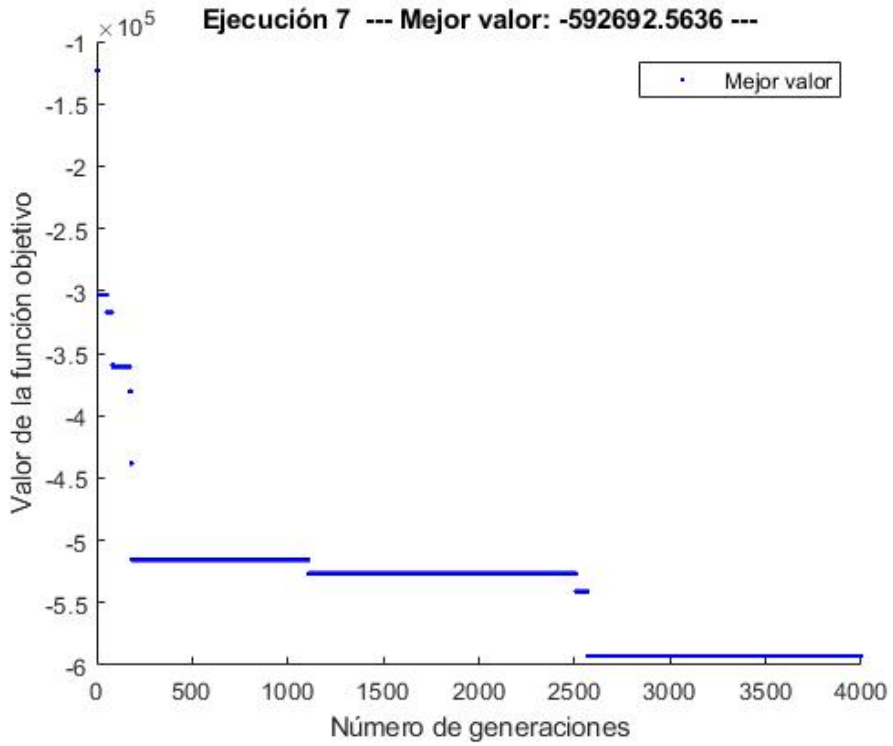


Figura 6.47 Representación de los mejores valores para la séptima ejecución del algoritmo en el problema *GTOC1*.

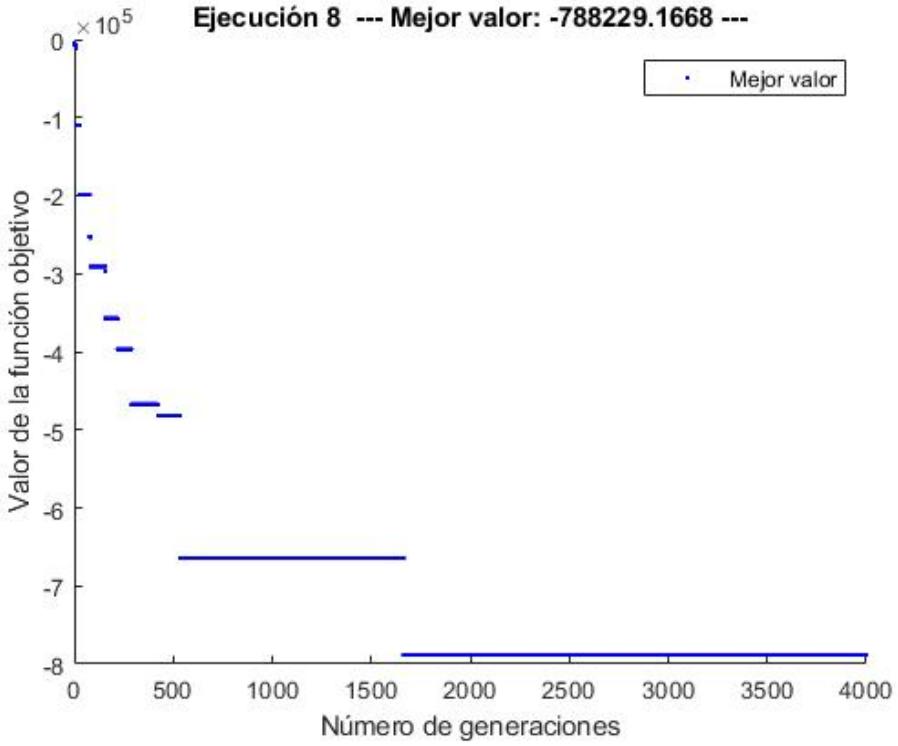


Figura 6.48 Representación de los mejores valores para la octava ejecución del algoritmo en el problema *GTOC1*.

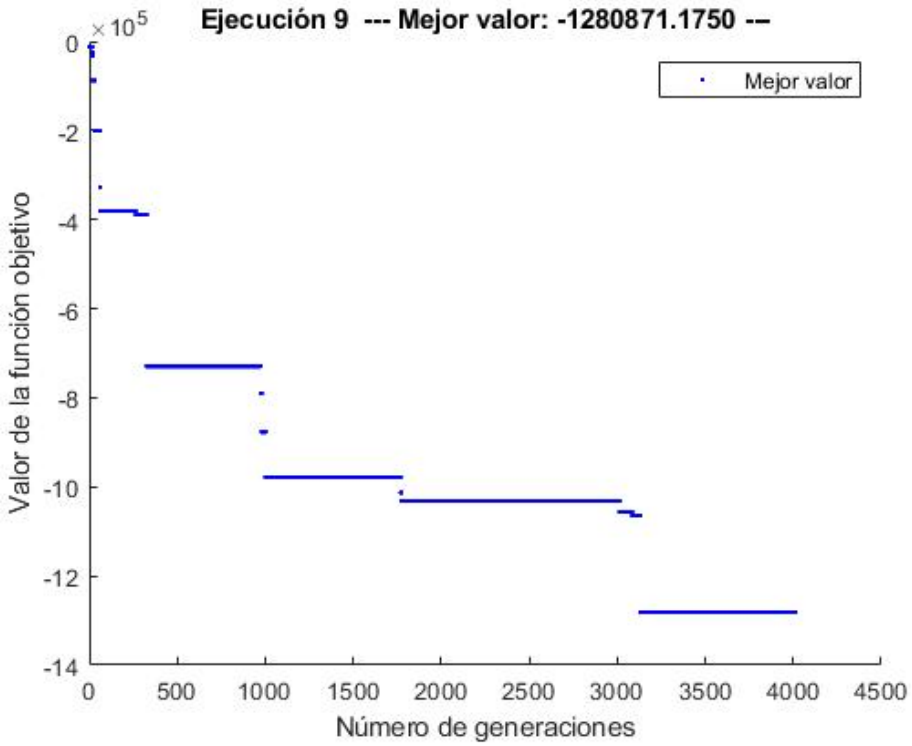


Figura 6.49 Representación de los mejores valores para la novena ejecución del algoritmo en el problema *GTOC1*.

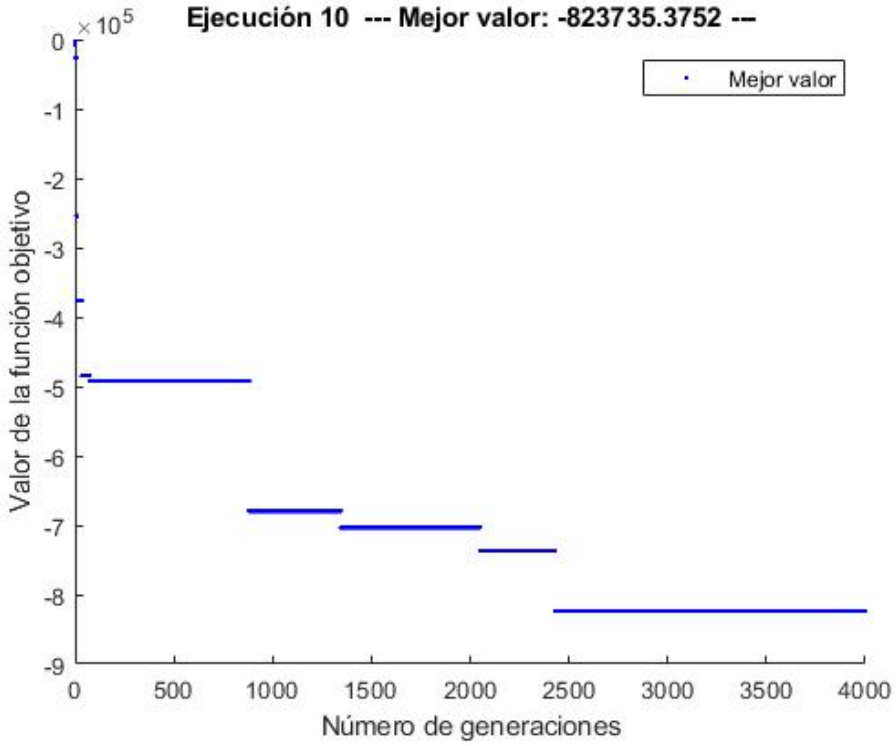


Figura 6.50 Representación de los mejores valores para la décima ejecución del algoritmo en el problema *GTOC1*.

Tabla 6.8 Resultados obtenidos durante las 10 ejecuciones junto con los tiempos de ejecución.

EJECUCIONES	J (julios)	Tiempo (min)
1	-878666.2302	29.9553
2	-707329.7736	29.7729
3	-947645.9178	30.2757
4	-850538.4341	29.9685
5	-1056350.6829	30.0059
6	-1019794.0846	30.0014
7	-592692.5636	30.0936
8	-788229.1668	30.6431
9	-1280871.1750	30.5965
10	-823735.3752	30.3293

No van a repetirse de nuevo los mismos comentarios acerca del funcionamiento del algoritmo puesto que son idénticos a los realizados para el problema *Cassini1*. Únicamente resaltar que el mejor resultado obtenido, tras unas 5.0274 horas, es de 1280871.1750 julios, menor que el encontrado por el algoritmo genético luego da una idea de que para este problema, algo más complejo, la actuación del algoritmo de optimización por enjambre es peor que el genético.

Algoritmo mixto

Para la aplicación del algoritmo mixto, se utilizarán los mismos parámetros que los anteriormente usados para cada algoritmo por separado. Manteniendo también el límite de 5000 generaciones utilizadas para resolver el problema *Cassini1*. Realizando 10 ejecuciones del algoritmo para poder sacar conclusiones sobre las tendencias de los resultados, se obtienen las figuras que se insertan seguidamente.

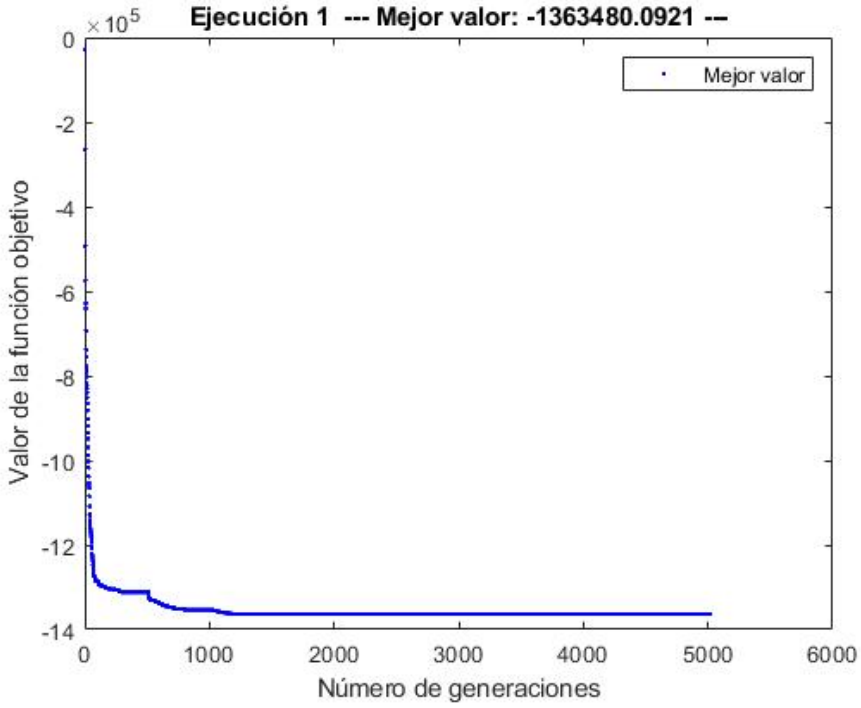


Figura 6.51 Representación de los mejores valores para la primera ejecución del algoritmo mixto en el problema *GTOC1*.

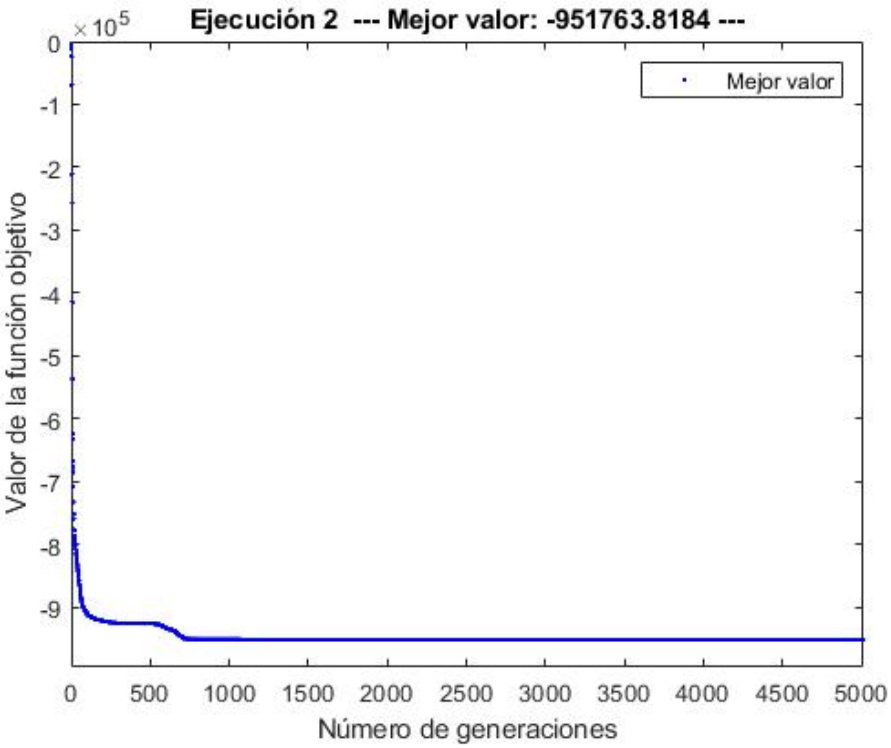


Figura 6.52 Representación de los mejores valores para la segunda ejecución del algoritmo mixto en el problema *GTOC1*.

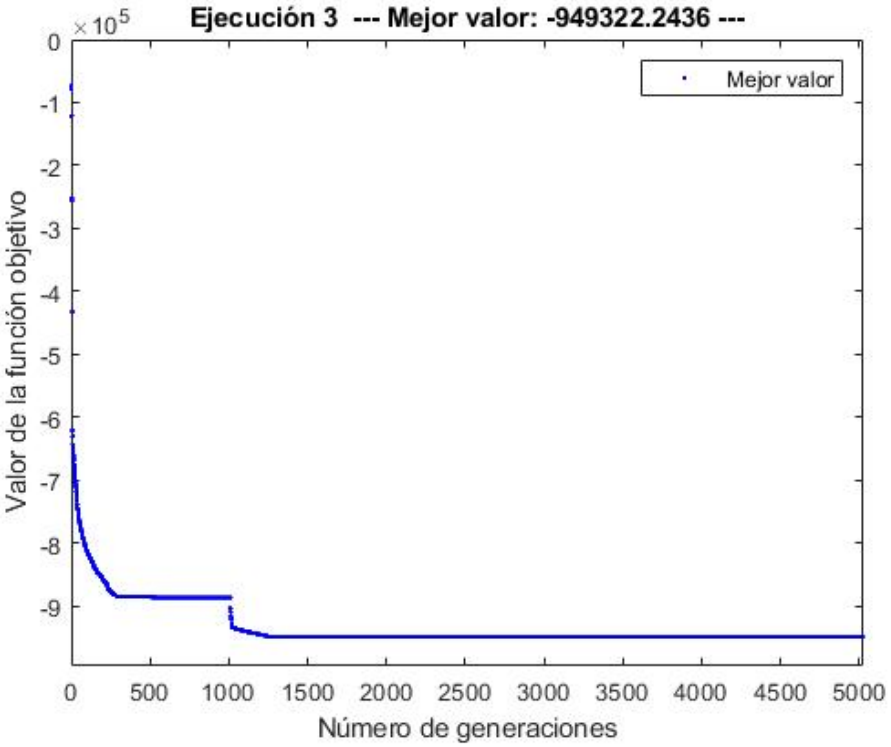


Figura 6.53 Representación de los mejores valores para la tercera ejecución del algoritmo mixto en el problema *GTOC1*.

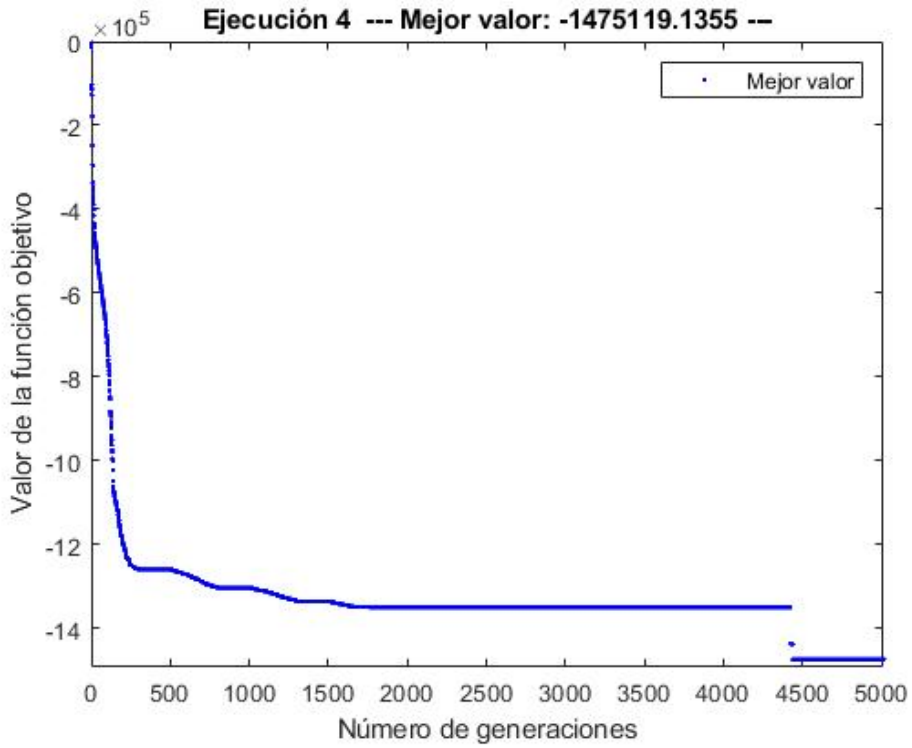


Figura 6.54 Representación de los mejores valores para la cuarta ejecución del algoritmo mixto en el problema *GTOC1*.

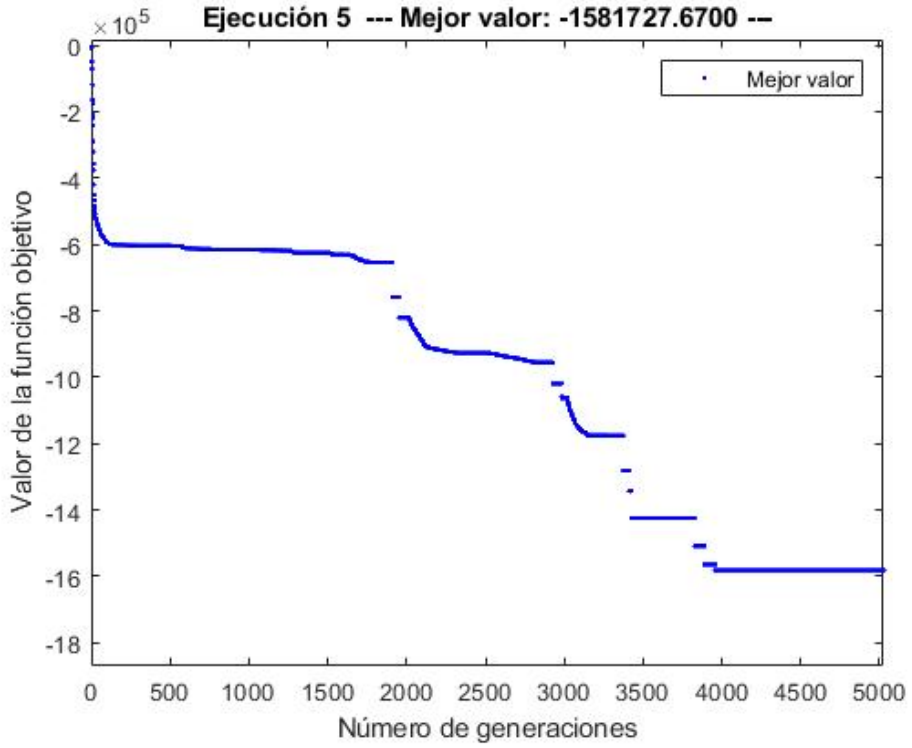


Figura 6.55 Representación de los mejores valores para la quinta ejecución del algoritmo mixto en el problema *GTOC1*.

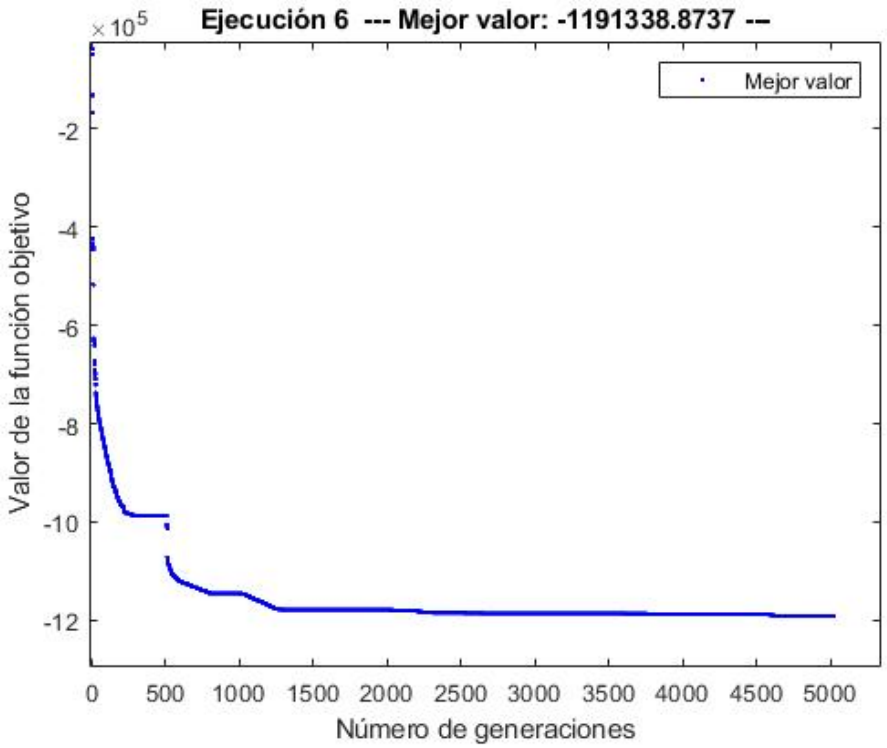


Figura 6.56 Representación de los mejores valores para la sexta ejecución del algoritmo mixto en el problema *GTOC1*.

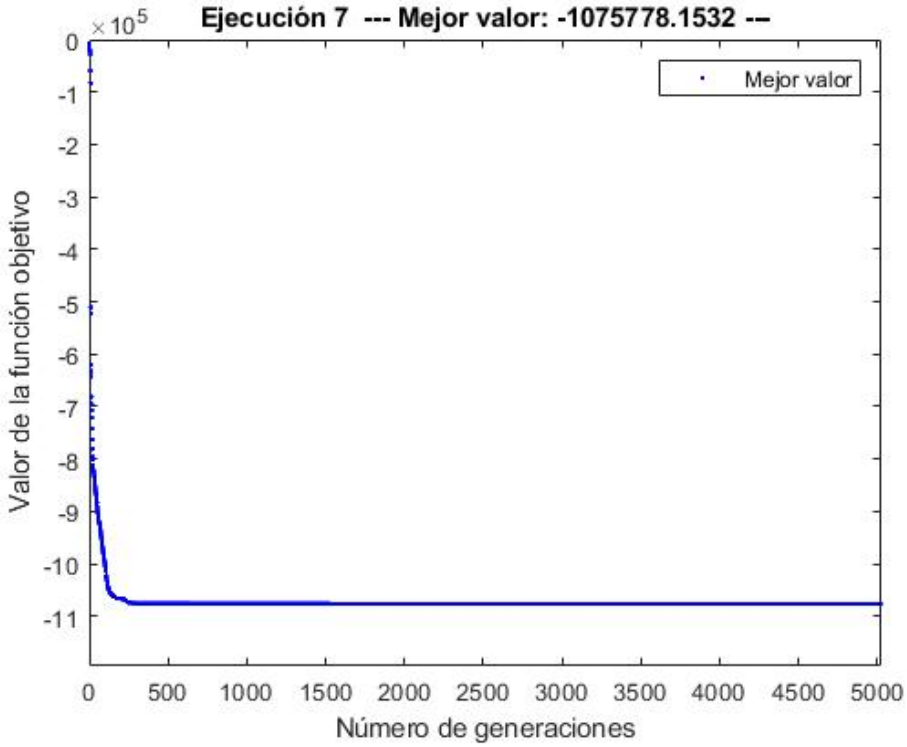


Figura 6.57 Representación de los mejores valores para la séptima ejecución del algoritmo mixto en el problema *GTOC1*.

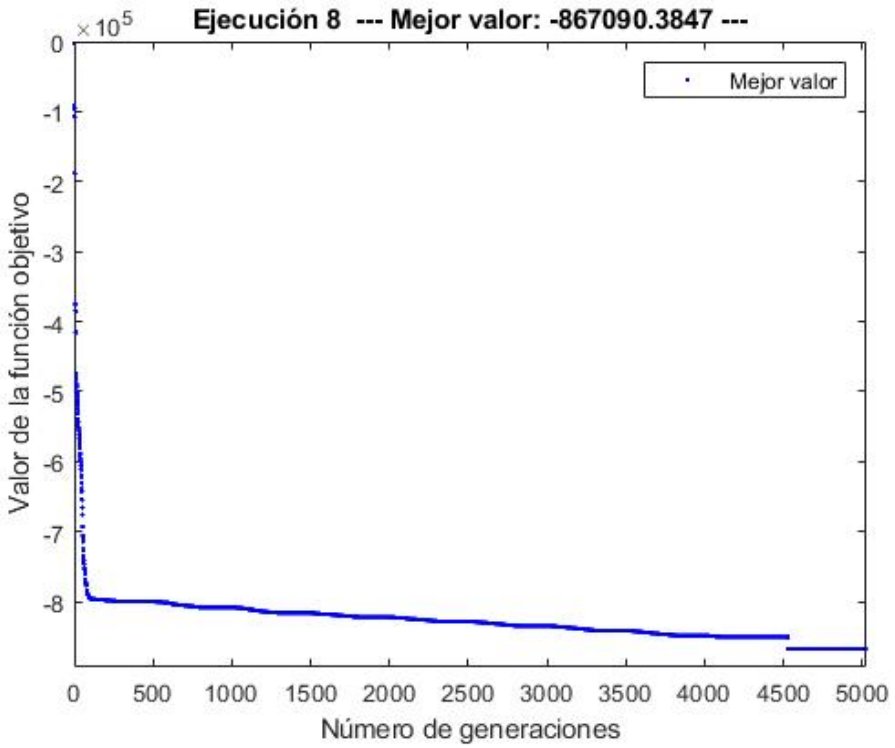


Figura 6.58 Representación de los mejores valores para la octava ejecución del algoritmo mixto en el problema *GTOC1*.

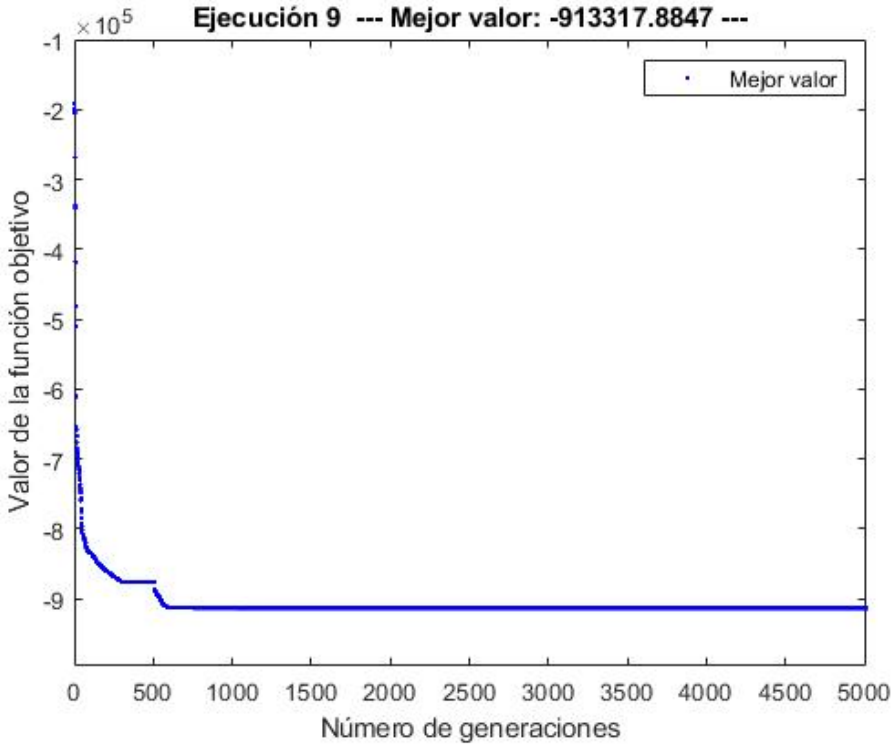


Figura 6.59 Representación de los mejores valores para la novena ejecución del algoritmo mixto en el problema *GTOC1*.

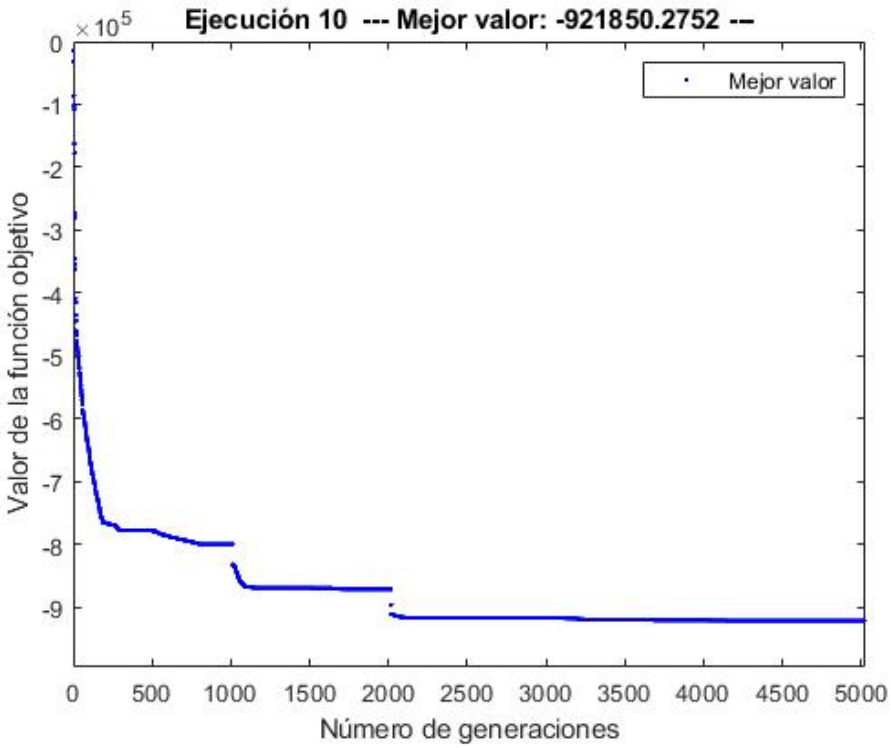


Figura 6.60 Representación de los mejores valores para la décima ejecución del algoritmo mixto en el problema *GTOC1*.

Tabla 6.9 Resultados obtenidos durante las 10 ejecuciones junto con los tiempos de ejecución.

EJECUCIONES	J	Tiempo (min)
1	-1363480.0921	36.3763
2	-951763.8184	36.4731
3	-949322.2436	36.3410
4	-1475119.1355	36.7127
5	-1581727.6700	36.9182
6	-1191338.8737	36.1517
7	-1075778.1532	38.2569
8	-867090.3847	36.3235
9	-913317.8848	36.3954
10	-921850.2752	36.3941

En este caso también se observan comportamientos parecidos a los ya comentados para el problema anterior, luego puede extraerse que el algoritmo mixto vuelve a funcionar adecuadamente para este problema. Además, este algoritmo ha logrado encontrar una solución mucho mejor que las obtenidas anteriormente, 1581727.6700 julios, para un tiempo relativamente pequeño, 6.1055 horas, por lo que el comportamiento del algoritmo es excepcional.

Comparativa y lógica de resultados

Para saber cómo de cercanos al óptimo son los resultados hallados, se han introducido junto con los publicados por la ESA [4] en la Tabla 6.10. Si se observa dicha tabla, se deduce que tanto el algoritmo genético como el mixto han logrado resultados aceptables por su similitud al óptimo

encontrado por los investigadores M. Schlueter y M. Gerdts de la Universidad de Birmingham. Hay que hacer mención especial al algoritmo mixto que, incluso, ha permitido mejorar el resultado calculado por los creadores de los modelos que se están optimizando, Tamas Vinko y Dario Izzo (dos investigadores de la ESA). Es fácilmente deducible que el algoritmo que mejor ha optimizado el problema planteado es el algoritmo mixto tanto por el tiempo que emplea en cada ejecución como por los resultados obtenidos en ellas. Además, para este problema el algoritmo de optimización por enjambre de partículas se ha quedado muy lejos de los mejores valores publicados aunque si se compara con el resultado del algoritmo aleatorio no cabe duda de que sí que ha funcionado pese a que esas 10 ejecuciones no han sido suficientes para encontrar una solución más cercana al óptimo.

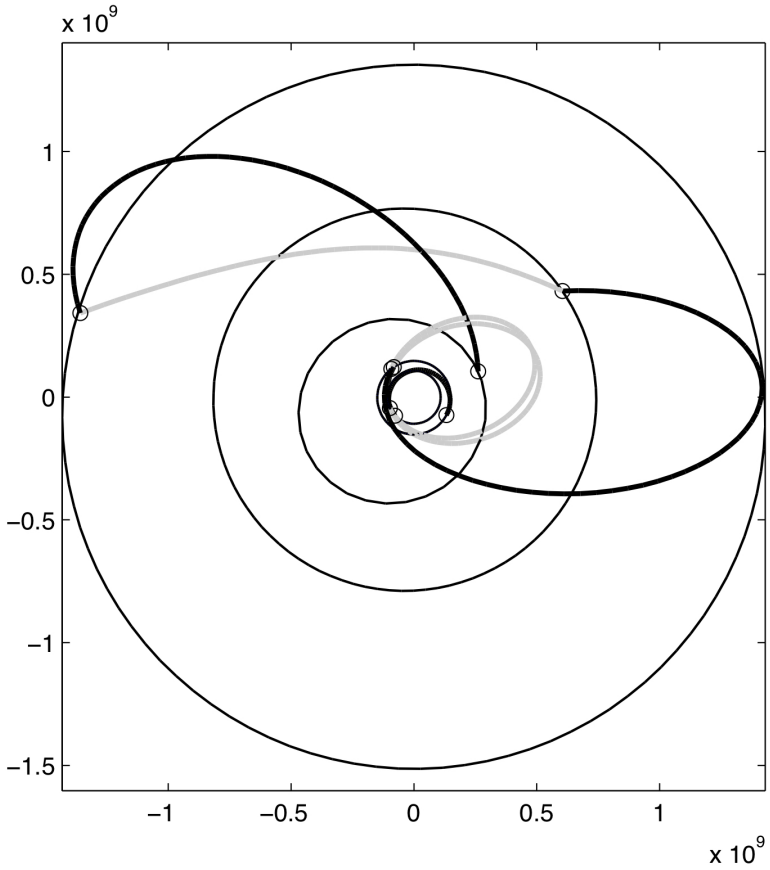


Figura 6.61 Trayectoria seguida para la mejor solución encontrada del problema GTOC1. Extraída de [15]. Órbitas representadas: Venus, Tierra, asteroide, Júpiter, Saturno.

Tabla 6.10 Mejores resultados obtenidos en el trabajo y publicados en la página de la ESA.

RESULTADOS DEL TRABAJO		RESULTADOS PUBLICADOS	
ALGORITMO	MEJOR VALOR	ALGORITMO	MEJOR VALOR
GA	1508657.7674 julios	PaGMO	1580599 julios
PSO	1280871.1749 julios	MIDACO	1581950 julios
MIXTO	1581727.6700 julios	-	-
ALEATORIO	459722.4627 julios		

Para el cálculo de los tiempos de duración de la misión según el resultado, se han tenido en cuenta los tiempos de vuelo en las trayectorias interplanetarias proporcionados por el vector de decisión

obtenidos durante la optimización. En la Tabla 6.11, se incluyen los tres vectores de decisión que generan como resultado los mejores valores obtenidos por cada algoritmo.

Tabla 6.11 Componentes de los vectores de decisión que dan lugar a los mejores resultados de la función objetivo hallados.

RESULTADO	COMPONENTES DEL VECTOR DE DECISIÓN			
J (x) (julios)	t_0 (MJD2000)	T_1 (Días)	T_2 (Días)	T_3 (Días)
1508657.7674	5660.654094	171.510697	1081.054980	72.248978
1280871.1749	7164.879275	247.859876	46.722715	861.963186
1581727.6700	6809.449827	169.443442	1079.493577	56.513105

RESULTADO	COMPONENTES DEL VECTOR DE DECISIÓN			
J (x) (julios)	T_4 (Días)	T_5 (Días)	T_6 (Días)	T_7 (Días)
1508657.7674	1787.804792	4090.971046	1062.386384	5021.924359
1280871.1749	1199.799335	3420.737244	1088.665342	4919.007154
1581727.6700	1044.054080	3823.417933	1043.058188	3393.629496

Los tiempos totales que duraría la misión de desvío del asteroide, que pueden calcularse sumando los tiempos de vuelos intermedios (T_1, ..., T_7), se han recogido en la Tabla 6.12. Puesto que no se tiene una referencia real, al ser un problema ideado simplemente para su optimización, solamente se puede comentar sobre los resultados que es destacable que el caso donde mayor cambio de semieje mayor de la órbita se conseguiría es para el que menos duraría la misión.

Como ya se mencionó para el problema *Cassini1*, esto no demuestra que esta sea la opción óptima del desarrollo de la misión puesto que simplemente se ha optimizado uno de los parámetros influyentes. Por ejemplo, habría que realizar un análisis del combustible gastado para comprobar la viabilidad de la misión

Tabla 6.12 Tiempos de duración de la misión según el resultado.

RESULTADO	DURACIÓN DE LA MISIÓN
J (x) (julios)	T_{total} (Años)
1508657.7674	36.4052
1280871.1750	32.2870
1581727.6700	29.0674

6.2 Problema con maniobras en espacio profundo

6.2.1 Messenger

Se ha visto en el Capítulo 4 que el objetivo de este problema vuelve a ser la minimización de los impulsos realizados durante la misión a Mercurio. A diferencia con los problemas anteriores, en este caso sí que se utilizan maniobras en espacio profundo incrementando el número de variables del vector de decisión, respecto de las cuales se realiza la optimización, hasta 18. Debido a este incremento en variables, no va a poder seguirse el mismo método de análisis que para los problemas anteriores ya que no van a obtenerse resultados buenos en poco tiempo de ejecución del algoritmo.

Por tanto, van a realizarse ejecuciones de cada algoritmo con un mayor número de generaciones o iteraciones de modo que sea posible encontrar soluciones adecuadas conociendo las limitaciones de tiempo y equipos que se tienen. Respecto a los parámetros usados en cada algoritmo, se usan todos los hallados en el Capítulo 5 salvo el tamaño de población y de enjambre para los algoritmos genético y de enjambre de partículas respectivamente.

Una vez se establecen los parámetros, se realizan 5 ejecuciones del problema con cada uno de los algoritmos presentados en el trabajo para comprobar el comportamiento de cada uno de ellos en este problema y los resultados obtenidos con los publicados por la ESA [4]. Se ha elegido este número de ejecuciones teniendo en cuenta la complejidad del problema y el tiempo necesario para cada una de ellas (casi un día).

Algoritmo genético

Como se ha comentado, los parámetros a utilizar para el análisis son los mismos que para el resto de problemas salvo el tamaño de población. Nótese que ahora el tamaño es de 10000 individuos con el fin de tener la capacidad de una búsqueda más amplia ya que la función objetivo es mucho más compleja que para los problemas anteriores. Recordando los parámetros, se tienen:

- Factor de mutación: 0.02
- Parámetro de cruce: 0.75
- Tamaño de la población: 10000 individuos
- Población de la élite: 1 % del total de la población

En cuanto al número de generaciones límite, también se ha aumentado hasta 35000 generaciones, por tanto, es lógico el incremento de tiempo por ejecución que aparece respecto a las ejecuciones realizadas en los problemas anteriores.

A continuación, se muestran cinco pares figuras correspondientes a las cinco ejecuciones del algoritmo: la primera es una visión general de los resultados obtenidos y la segunda es un zoom para ver como ha ido variando el mejor valor con el paso de las generaciones. De nuevo, los mejores valores de cada generación se representan en puntos negros, mientras que la media de los valores de la función objetivo de los individuos de cada generación en puntos azules.

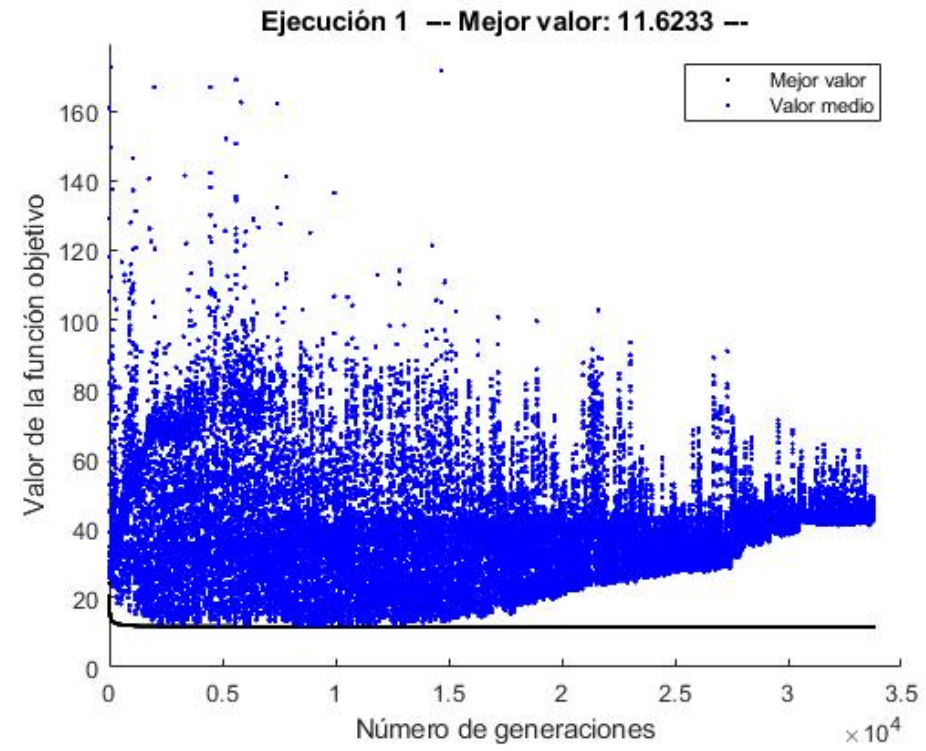
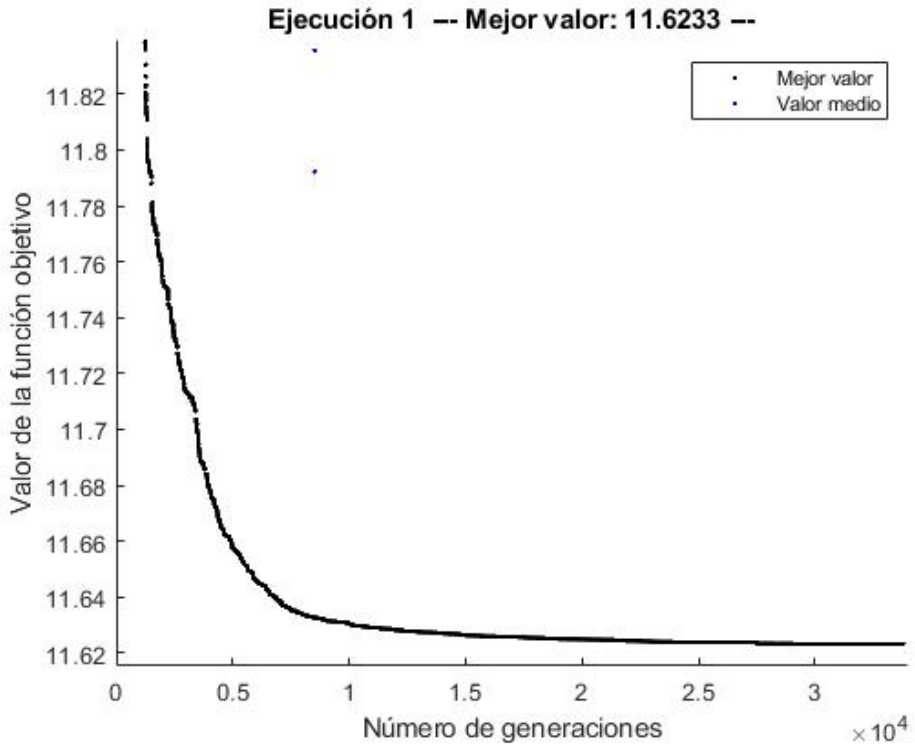


Figura 6.62 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la primera ejecución del algoritmo genético en el problema *Messenger*.



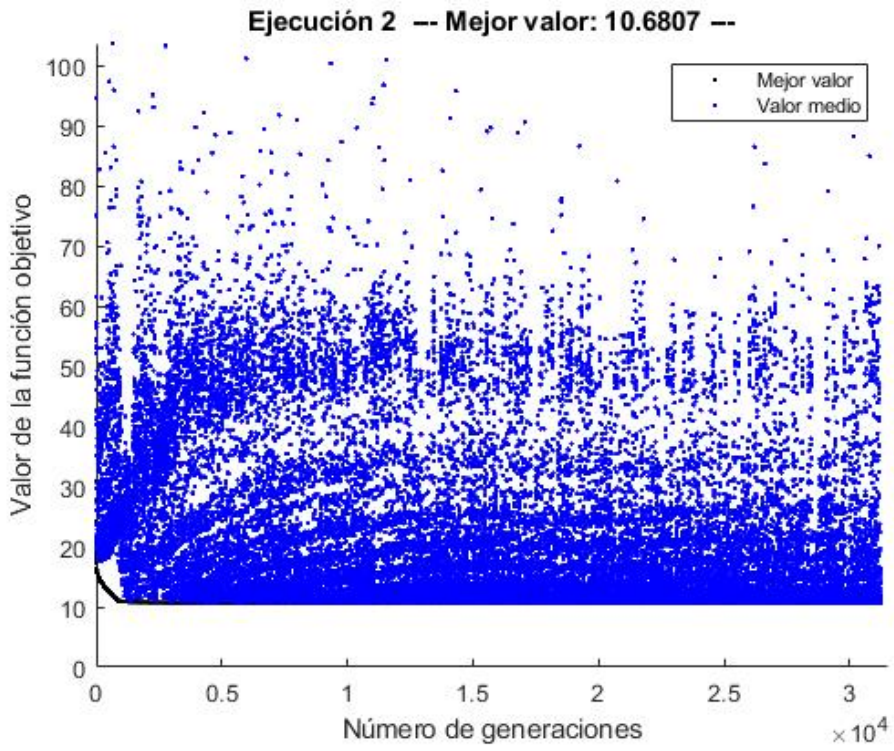
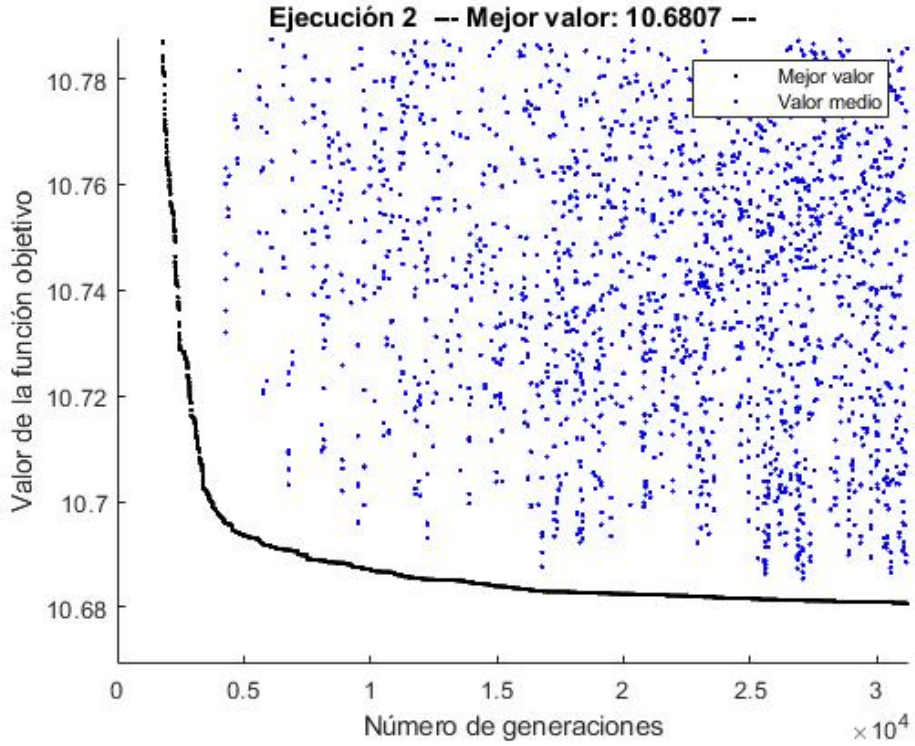


Figura 6.63 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la segunda ejecución del algoritmo genético en el problema *Messenger*.



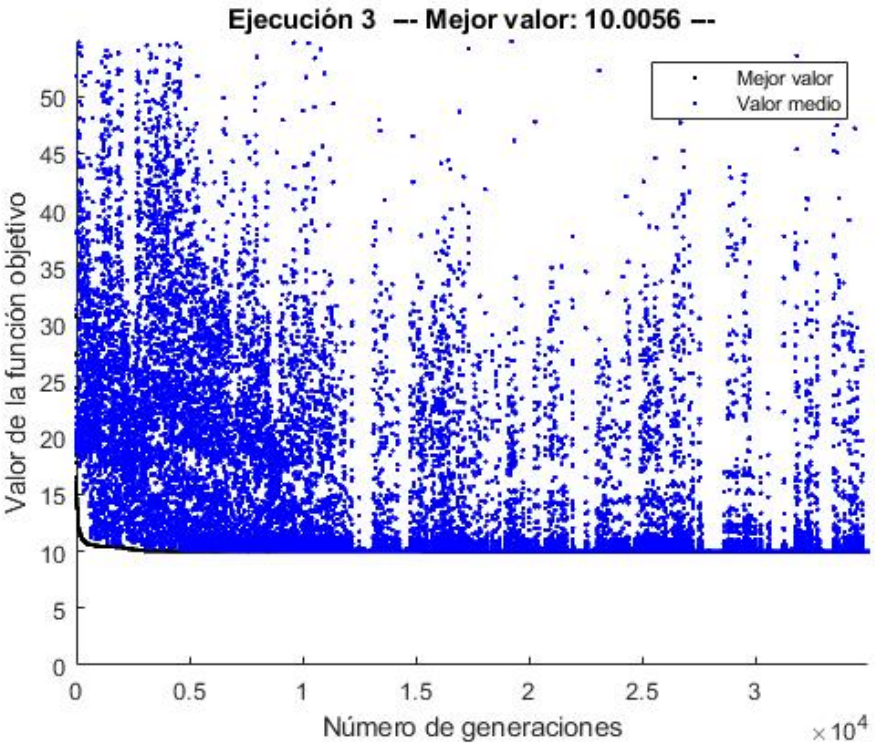
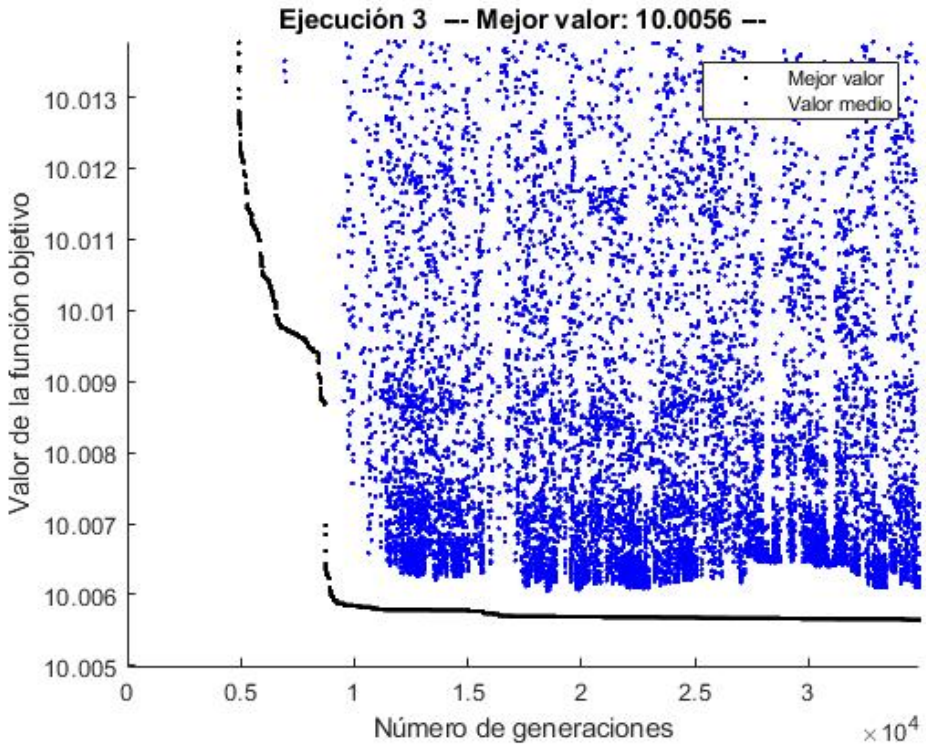


Figura 6.64 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la tercera ejecución del algoritmo genético en el problema *Messenger*.



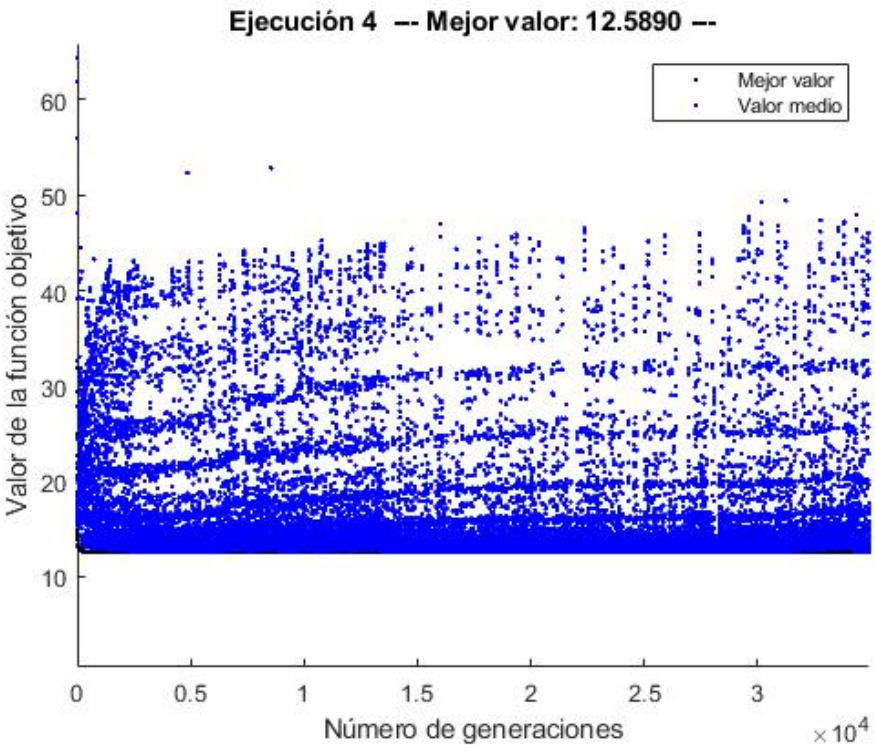
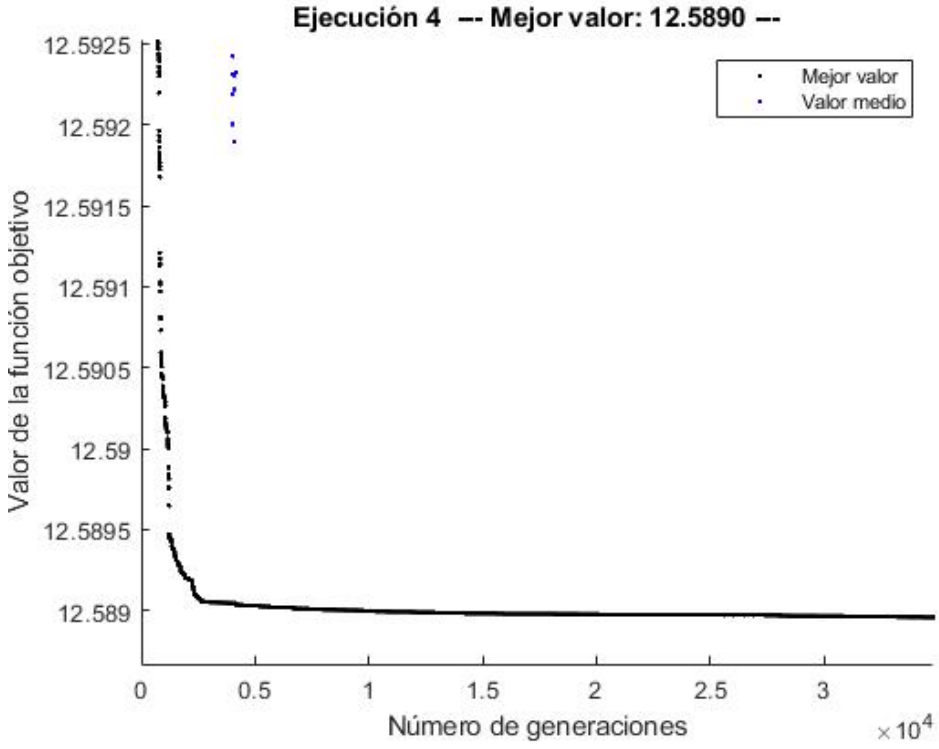


Figura 6.65 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la cuarta ejecución del algoritmo genético en el problema *Messenger*.



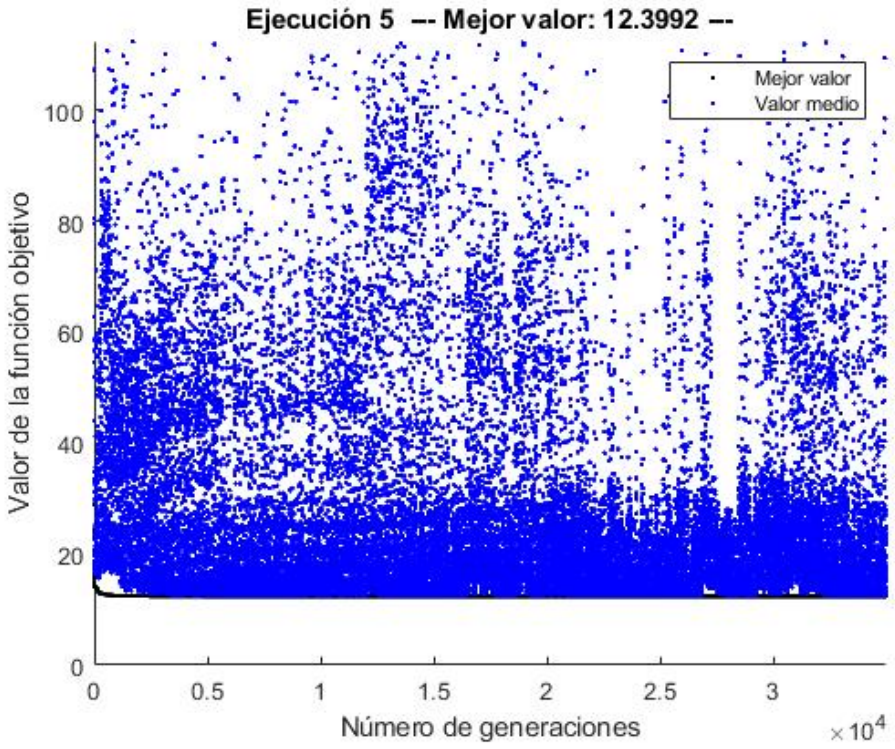


Figura 6.66 Representación de los mejores valores (puntos negros) y de la media (puntos azules) por generación para la quinta ejecución del algoritmo genético en el problema *Messenger*.

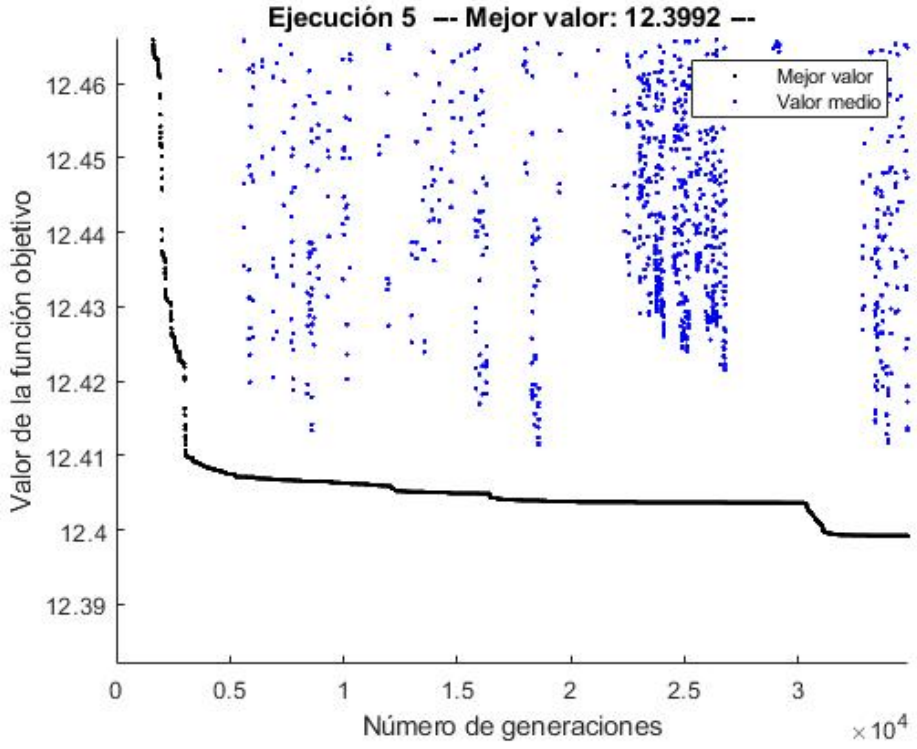


Tabla 6.13 Resultados obtenidos durante las 5 ejecuciones junto con los tiempos de ejecución.

EJECUCIONES	J (km/s)	Tiempo (horas)
1	11.6233	22.9682
2	10.6807	21.3907
3	10.0056	23.7467
4	12.5890	22.4163
5	12.3992	22.3721

En las primeras figuras mostradas para cada generación se observa como en ninguna de ellas la media llega a parecerse al mejor valor en las generaciones finales, esto se debe principalmente a la complejidad de la función objetivo. Si se evalúa la función para dos vectores de decisión idénticos excepto alguna modificación en una de las variables puede verse la gran diferencia de resultados que generan, por ello con el paso de las generaciones y los cambios que sufren los diferentes individuos es muy poco probable que la media de todos los de una generación se parezcan al mejor valor. También es una muestra acerca de la búsqueda que realiza el algoritmo con los nuevos individuos, puesto que la media difiere del mejor valor cabe la posibilidad que alguna combinación de individuos con un valor asociado lejano al óptimo de la función objetivo genere un individuo mejor que el encontrado hasta esa generación. Por esto último, puede verse en las segundas figuras de cada ejecución como siempre se tiene un comportamiento descendente en los puntos que representan el mejor valor.

Destacar el comportamiento de la media en la primera ejecución, Figura 6.62, en la que se observa claramente un aumento con el paso de las generaciones. Si bien no puede conocerse los motivos exactos que dan lugar a dicho comportamiento, puede deducirse que aparece debido a la combinación entre mutación y cruce de cromosomas que generan nuevos individuos parecidos entre ellos y muy diferentes del mejor de todos ellos.

El mejor resultado obtenido tras las 5 ejecuciones y un total de 4.7039 días de cálculos con MATLAB es 10.0056 km/s que a priori parece estar lejos del mejor resultado obtenido por los investigadores F. Biscani, M. Rucinski y D. Izzo de la ESA (8.630 km/s), pero que en comparación el resultado que arroja un algoritmo genético supone una mejora drástica como se muestra en la sección de comparación final. Finalmente añadir que puede verse en las figuras que presentan el zoom a la curva de mejores valores como, para las generaciones finales el algoritmo, sigue mejorando el resultado. Esto permite concluir que si se estableciera un límite de generaciones mayor se obtendrían mejores resultados que los presentados, pero sería inviable para el trabajo desarrollado por los enormes tiempos que conllevaría.

Algoritmo de optimización por enjambre de partículas

A continuación se muestran de nuevo los parámetros a utilizar en la optimización de los cuales únicamente se ha aumentado el tamaño del enjambre siendo ahora de 10000 partículas para poder abarcar más espacio de búsqueda en cada iteración del algoritmo:

- Factor de inercia: 1
- Coeficiente de atracción al mejor personal: 0.75
- Coeficiente de atracción al mejor global: 0.25
- Tamaño del enjambre: 10000 partículas

También ha cambiado el número de iteraciones máximas que van a realizarse, se ha aumentado hasta 12000 para que la ejecución tenga el máximo de iteraciones posibles para encontrar un buen

resultado sin que el tiempo de la misma sea excesivo. Los resultados de las ejecuciones se muestran seguidamente.

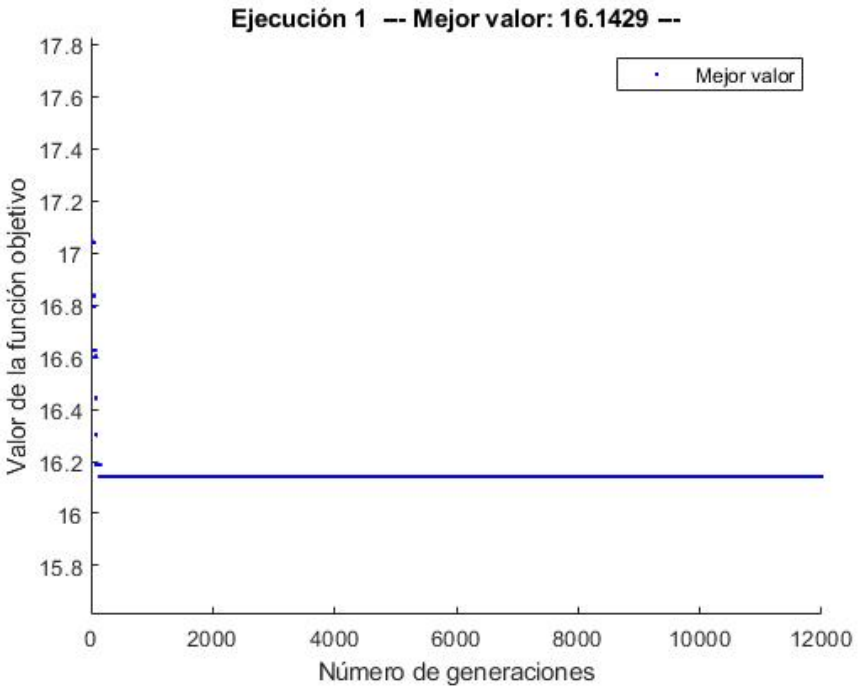


Figura 6.67 Representación de los mejores valores para la primera ejecución del algoritmo en el problema *Messenger*.

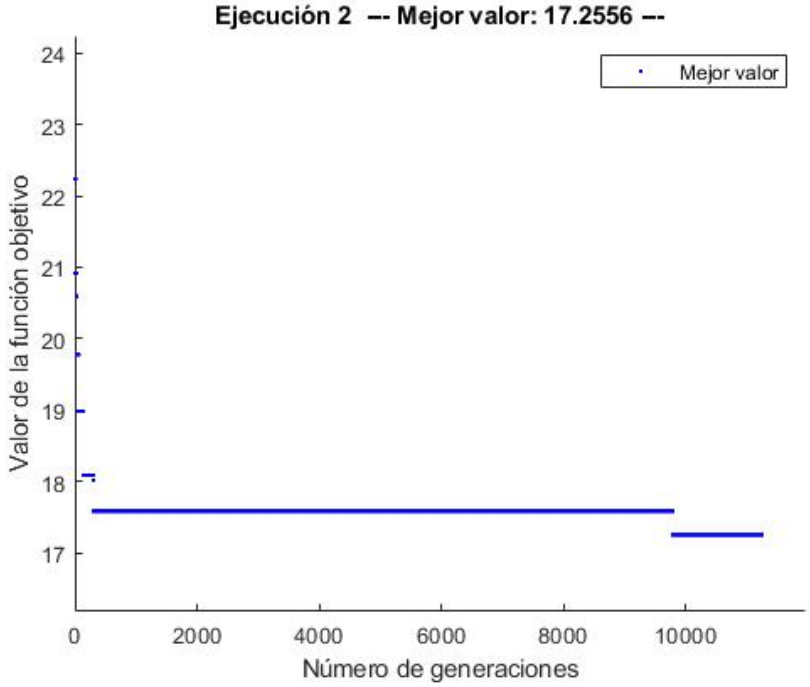


Figura 6.68 Representación de los mejores valores para la segunda ejecución del algoritmo en el problema *Messenger*.

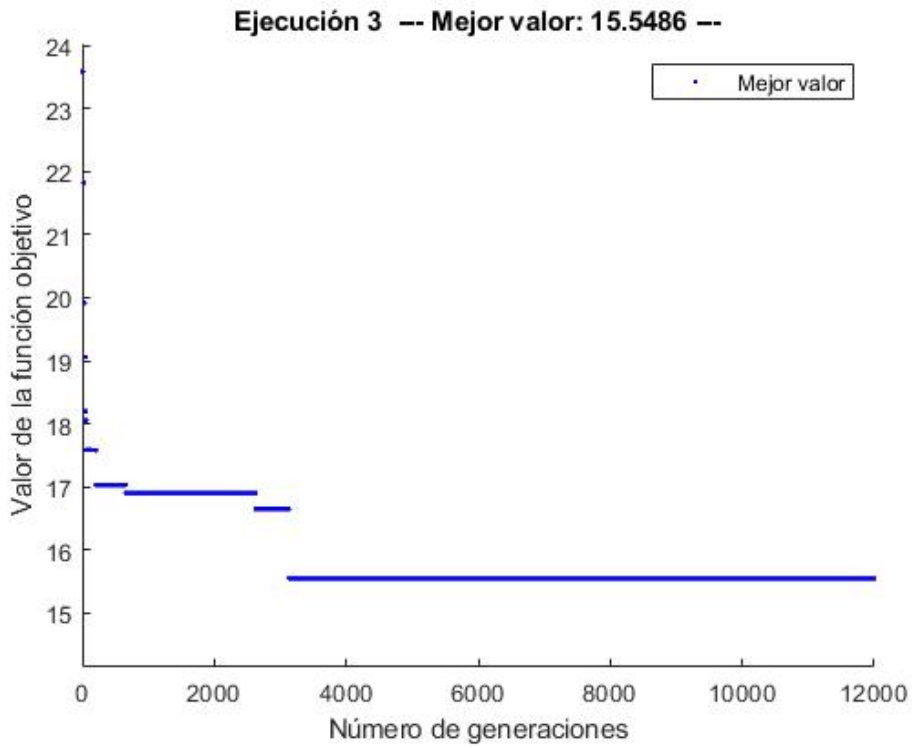


Figura 6.69 Representación de los mejores valores para la tercera ejecución del algoritmo en el problema *Messenger*.

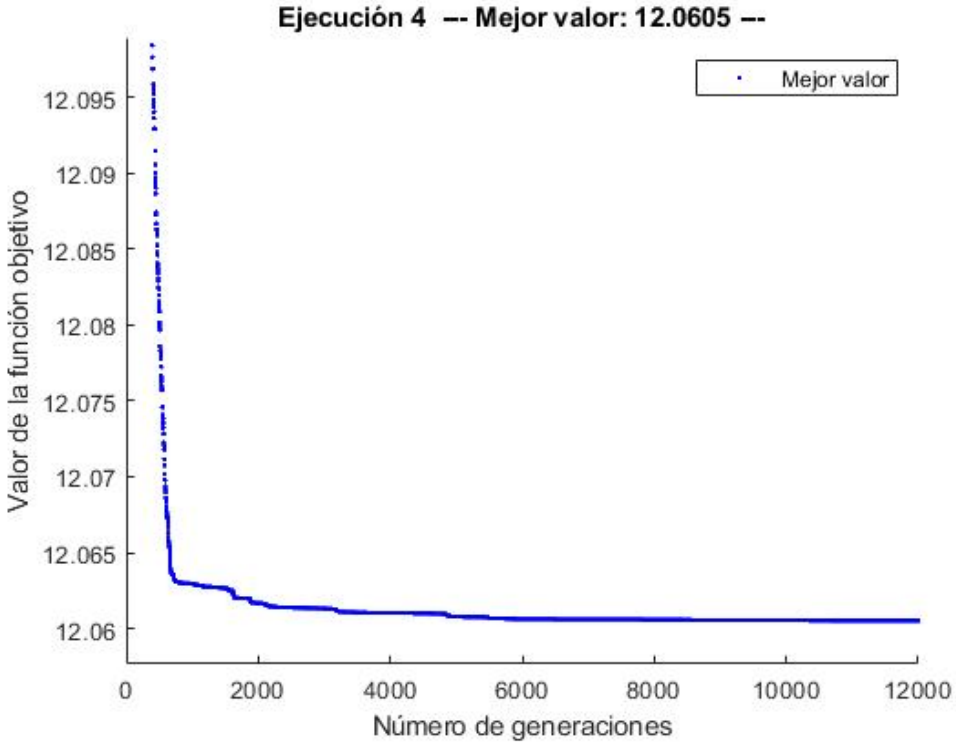


Figura 6.70 Representación de los mejores valores para la cuarta ejecución del algoritmo en el problema *Messenger*.

Tabla 6.14 Resultados obtenidos durante las 5 ejecuciones junto con los tiempos de ejecución.

EJECUCIONES	J (km/s)	Tiempo (horas)
1	16.1429	22.8852
2	17.2556	21.4857
3	15.5486	23.1602
4	12.0605	22.1486

Para este algoritmo vuelve a tenerse un comportamiento parecido al de los problemas anteriores, en gran parte de las iteraciones se observa que el algoritmo no mejora pero continúa buscando ya que, pese a permanecer horizontal la curva de puntos, existen algunas iteraciones en las que aparece un escalón de mejora de resultados.

En cuanto a los resultados obtenidos puede verse que están más alejados que los del algoritmo anterior del mínimo global encontrado por los investigadores de la ESA (8.630 km/s) ya que el mejor de todos ellos ha sido 12.0605 km/s tras 3.7367 días de análisis. Por tanto, puede concluirse que el algoritmo de optimización por enjambre de partículas no ha funcionado tan bien como para el problema *Cassini1* y puede plantearse como mejora futura la realización de un análisis de los parámetros de control específicamente para este problema con el fin de mejorar los resultados.

Algoritmo mixto

Haciendo uso de los parámetros comentados para cada algoritmo se aplica el algoritmo mixto al problema *Messenger* pero con un máximo de 12500 iteraciones combinando ambos algoritmos, es decir, 2.5 veces más que para los problemas anteriores. Esto se debe a que se ha pretendido hacer ejecuciones de un tiempo de ejecución de un día para ver si el funcionamiento de este método en la optimización del problema mejora o no a los algoritmos por separados. De igual manera se han realizado 5 ejecuciones cuyos resultados se muestran a continuación.

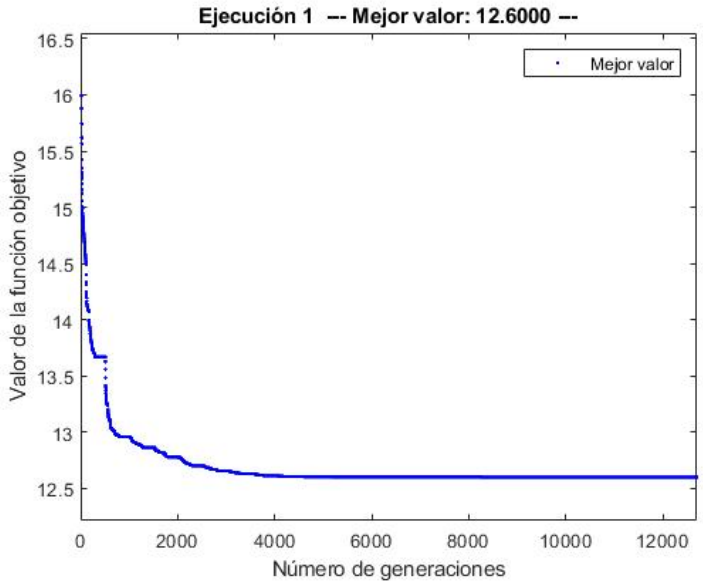


Figura 6.71 Representación de los mejores valores para la primera ejecución del algoritmo mixto en el problema *Messenger*.

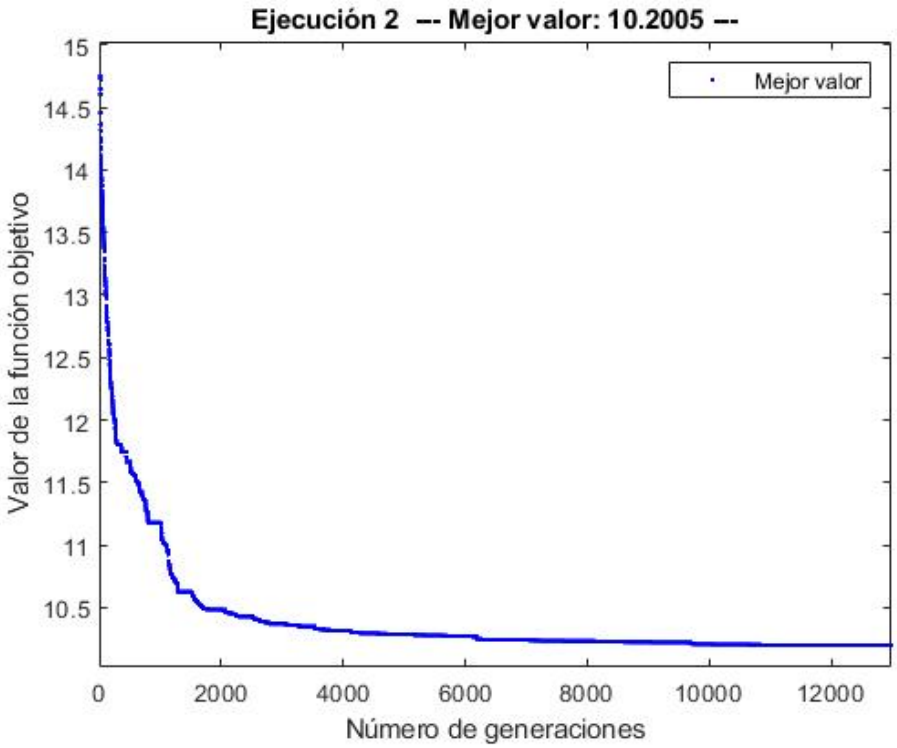


Figura 6.72 Representación de los mejores valores para la segunda ejecución del algoritmo mixto en el problema *Messenger*.

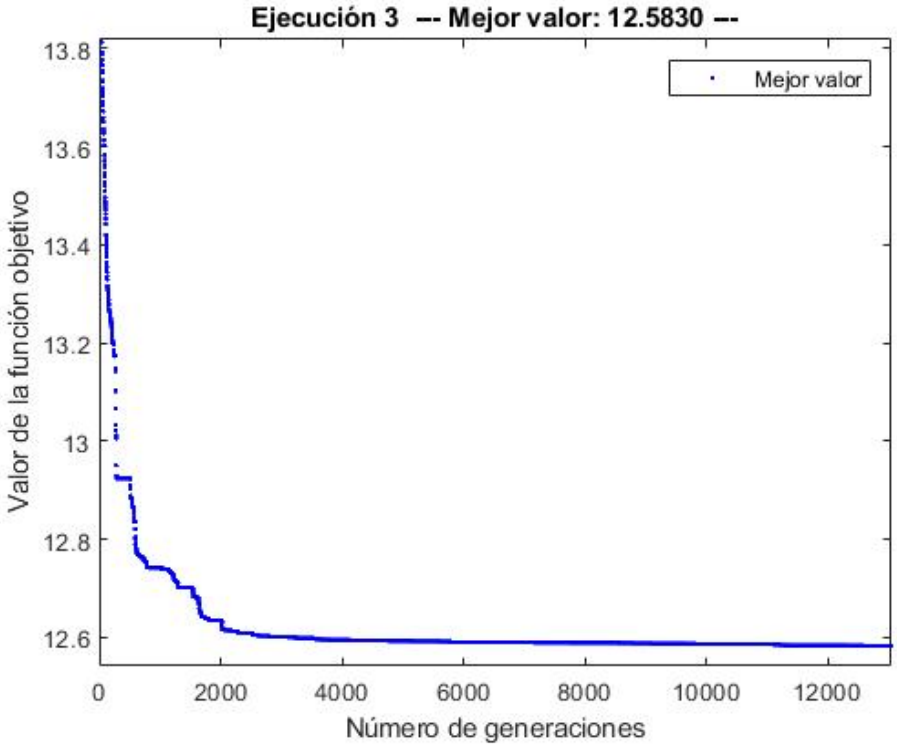


Figura 6.73 Representación de los mejores valores para la tercera ejecución del algoritmo mixto en el problema *Messenger*.

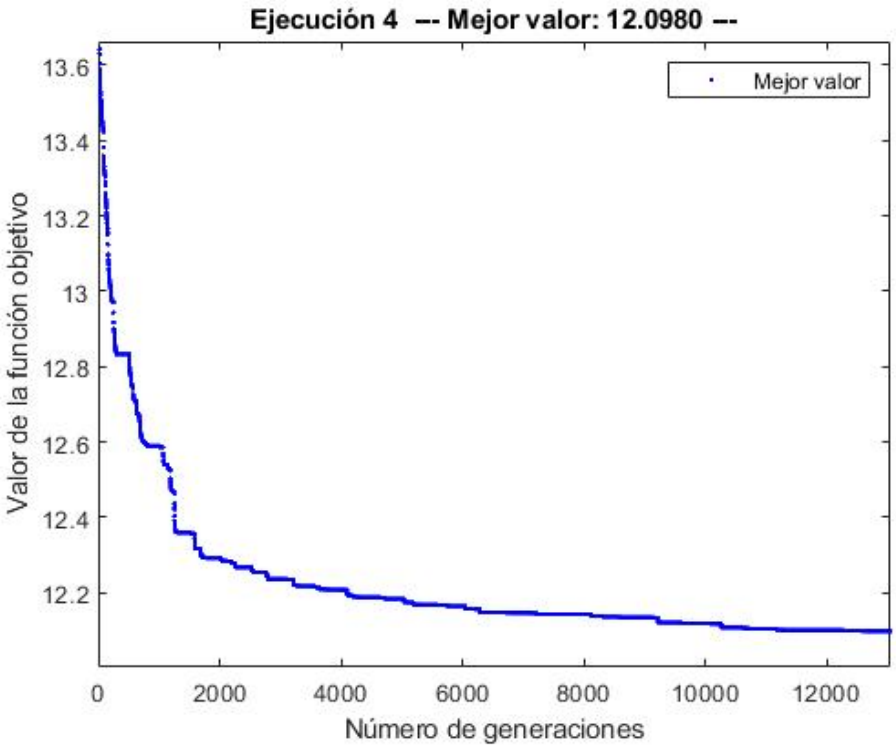


Figura 6.74 Representación de los mejores valores para la cuarta ejecución del algoritmo mixto en el problema *Messenger*.

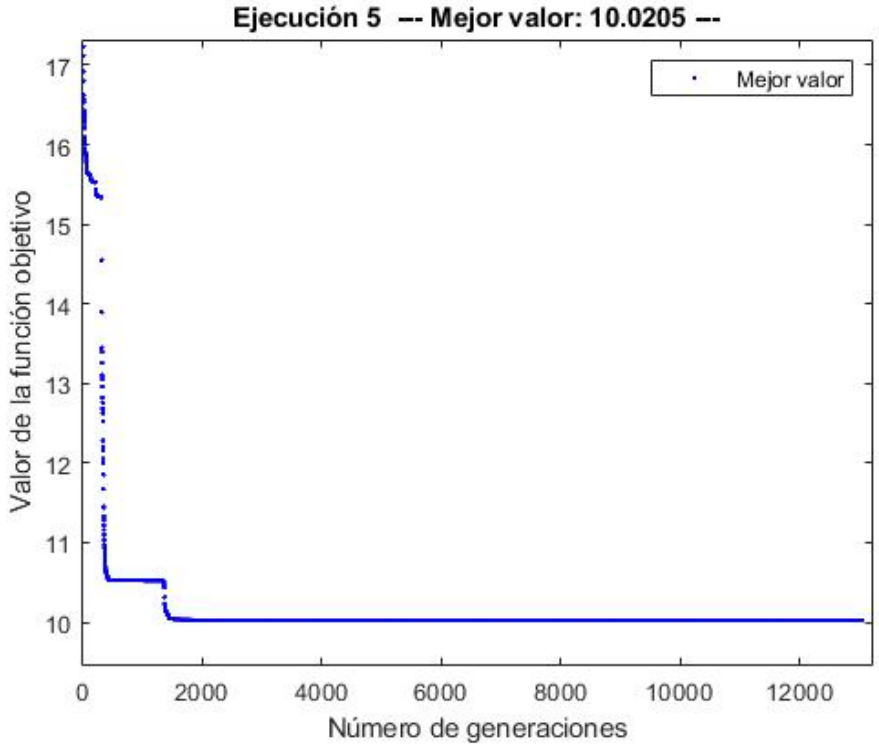


Figura 6.75 Representación de los mejores valores para la quinta ejecución del algoritmo mixto en el problema *Messenger*.

EJECUCIONES	J (km/s)	Tiempo (horas)
1	12.6000	23.2487
2	10.2005	23.7897
3	12.5830	23.3741
4	12.0980	23.0770
5	10.0205	22.9445

Se observa de nuevo en las figuras presentadas que el algoritmo mixto vuelve a tender hacia valores mejores con el paso de las iteraciones combinando el buen funcionamiento del algoritmo genético con la aplicación de la optimización por enjambre de partículas que funciona a modo de preparación para la siguiente aplicación del algoritmo genético. El mejor resultado obtenido tras 4.8514 días de optimización por este algoritmo ha sido 10.0205 km/s que vuelve a ser mucho mejor que el obtenido mediante el algoritmo aleatorio, luego la búsqueda se ha realizado de la manera adecuada. Para conseguir valores mejores habría que realizar ejecuciones de una duración mucho mayor.

Comparativa y lógica de resultados

Finalmente, se presenta en esta última sección del problema *Messenger* una comparativa de los resultados obtenidos por los tres algoritmos junto con los mínimos publicados en la página de la ESA [4].

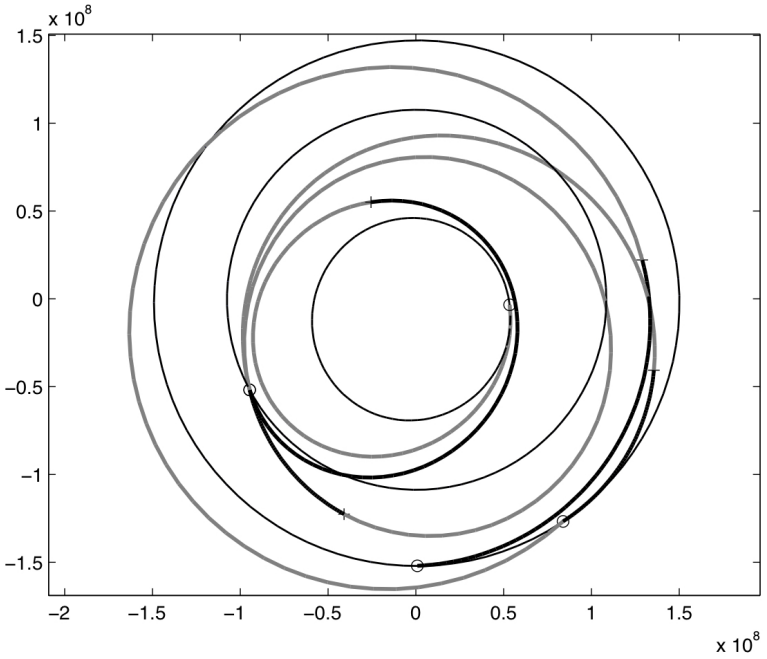


Figura 6.76 Trayectoria seguida para la mejor solución encontrada del problema Messenger. Extraída de [15]. Órbitas representadas: Mercurio, Venus, Tierra.

Tabla 6.15 Mejores resultados obtenidos en el trabajo y publicados en la página de la ESA.

RESULTADOS DEL TRABAJO		RESULTADOS PUBLICADOS	
ALGORITMO	MEJOR VALOR	ALGORITMO	MEJOR VALOR
GA	10.0056 km/s	DE	8.703 km/s
PSO	12.0605 km/s	MBH	8.631 km/s
MIXTO	10.0205 km/s	–	8.630 km/s
ALEATORIO	22.4685 km/s		

De la comparativa realizada en la Tabla 6.15, se extraen dos conclusiones claras:

- Primeramente, se observa que los resultados obtenidos no son muy parecidos a los publicados como ocurrió para los problemas sin maniobras en espacio profundo debido principalmente a la falta de equipos con la suficiente potencia de cálculo como pueden tener en la ESA, ya que los cálculos se han realizado con el ordenador portátil personal del autor del trabajo.
- En segundo lugar, si se comparan los resultados con el generado por el algoritmo aleatorio se observa que los tres algoritmos utilizados sí que han funcionado correctamente en su búsqueda del mínimo global, aunque sin fortuna en su encuentro.

Por último, adelantar que una posible mejora a realizar en este trabajo, comentada en el Capítulo 7, es la de llevar a cabo un estudio de los parámetros de control de los algoritmos similar al realizado pero aplicado específicamente al problema *Messenger*. Dicho estudio permitiría mejorar la conducta del algoritmo durante la optimización dando lugar a mejores soluciones para unos tiempos de ejecuciones parecidos.

7 Conclusiones y mejoras

7.1 Conclusiones

Los algoritmos genéticos están dotados de una gran aplicabilidad y eficiencia por el modo en el que se van creando las nuevas posibles soluciones del problema ya que no se ve afectado por las características de la función objetivo (continuidad, derivabilidad, etc.). De la misma manera los algoritmos de optimización por enjambre también pueden ser utilizados para el estudio de cualquier problema ya que la generación de las nuevas posiciones de las partículas en cada iteración vuelve a ser independiente de la forma de la función objetivo. Es por ello que estos métodos metaheurísticos suponen una ventaja clara frente a otros métodos de optimización más tradicionales, mediante los cuáles la optimización sería mucho más complicada e, incluso, imposible a veces debido a la complejidad de los problemas con los que se trata.

Otra de las ventajas principales de este tipo de algoritmo es que no requieren de una estimación inicial dada por el programador a partir de la cual comenzar a iterar y de la que depende el encontrar o no la solución buscada (típico de otros métodos tradicionales). Esto conlleva a que no sea necesario conocer a fondo un problema cuando se pretenda optimizar por algoritmos metaheurísticos, ya que estos algoritmos únicamente requieren la función objetivo y las posibles restricciones impuestas para alcanzar la solución óptima.

Respecto a los estudios realizados en este trabajo, destacar que, pese al tiempo que conlleva, el proceso de análisis de los parámetros de control de cada algoritmo ha resultado ser un éxito al menos en lo que respecta a los problemas sin maniobras en espacio profundo ya que se han logrado soluciones muy cercanas al óptimo encontrado por investigadores de la ESA o de otras grandes universidades para unos tiempos de ejecución de los algoritmos relativamente pequeños.

Finalmente comentar que la introducción del algoritmo mixto que combina los dos algoritmos metaheurísticos es útil en problemas con pocas variables en el vector de decisión, puesto que al crecer el número de variables respecto a las que se realiza la optimización las soluciones que este proporciona no son mejores que las del los algoritmos genéticos.

7.2 Mejoras futuras

El primero de los puntos a tener en cuenta en las posibles mejoras que pueden realizarse en futuros trabajos es el de la aplicación de la mutación adaptada para los algoritmos genéticos. Recuérdese que en este trabajo se ha hecho uso de la mutación con probabilidad uniforme, pero la adaptada supone una mejora considerable ya que tiene la capacidad de asociar una probabilidad a cada

individuo en función de su valor de la función objetivo. Es por ello que los mejores individuos de cada generación raramente se verán mutados evitando así perder los buenos resultados del algoritmo, mientras que para los peores de la generación la mutación será más probable consiguiendo, por tanto, una búsqueda más amplia por toda la región factible.

Debido a que solamente se ha realizado el estudio de parámetros de control para el problema *Cassini1* por la similitud entre las funciones objetivos que se tratan, se recomienda que una posible opción de mejora del funcionamiento de los algoritmos para los otros dos problemas estudiados es el llevar a cabo un análisis de los parámetros de control similar al que se ha expuesto en este trabajo. Tras el estudio exhaustivo se obtendrían los valores más adecuados de cada parámetro y mejoraría la actuación de los algoritmos en los problemas *GTOC1* y *Messenger*.

También convendría realizar una comparación entre tiempos de ejecución y resultados obtenidos por estos métodos metaheurísticos con los obtenidos por algunos métodos tradicionales para demostrar la mejora que suponen como se ha realizado con el algoritmo aleatorio.

Respecto al modelado de los problemas, podría complementarse este trabajo con un nuevo estudio de las misiones pero eliminando simplificaciones como el uso de una órbita de aparcamiento, que obligaría a modelar la fase de lanzamiento de la nave, o modelar las órbitas planetarias como son en la realidad en lugar de utilizar la aproximación circunferencial.

Apéndice A
Generaciones finales del algoritmo genético

En este apéndice se pretende mostrar brevemente y con un mayor detalle lo que ocurre cuando el algoritmo genético se estanca un mínimo local y la media de los valores de la función objetivo de todos individuos de cada generación comienza a parecerse cada vez más al menor valor encontrado. Pese a que a priori pueda parecer que el algoritmo deja de evaluar otras posibles soluciones porque la media se superpone al mejor valor, esto no es más que una consecuencia de la escala del eje *y* en la representación.

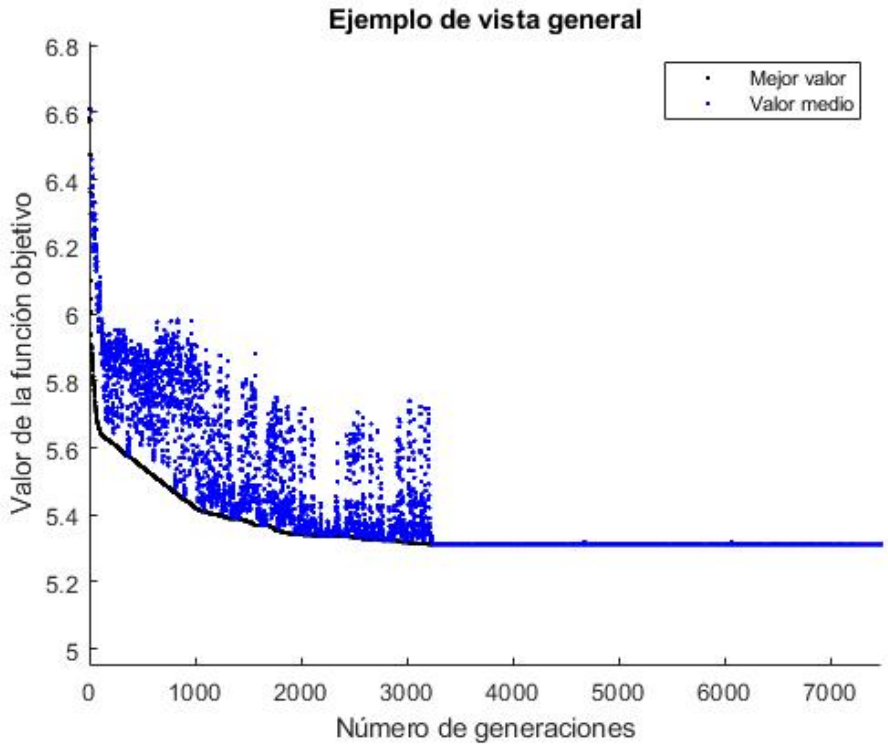


Figura A.1 Vista general para una ejecución cualquiera del algoritmo genético.

Si se hace zoom en la Figura A.1 de forma que se represente un intervalo menor en el eje *y* se verá

como, en realidad, la media sí que difiere en un cierto rango del mejor valor encontrado en cada generación. Esto se debe a que el propio cruce de los individuos progenitores genera descendencia que, en la mayoría de casos, lleva asociados peores valores de la función objetivo que el encontrado previamente. También es cierto que debido a la población de élite, que pasa de generación en generación sin verse modificada, en cada generación se tendrá un conjunto de individuos más parecidos entre ellos y más parecidos al mejor de todos ellos. Si el mejor de todos ellos no es el óptimo del problema, se dice que el algoritmo se ha estancado en un mínimo local y la única opción de encontrar una solución que mejore a la encontrada es que bien por mutación o bien en algún cruce de cromosomas determinado aparezca un individuo mejor de manera fortuita.

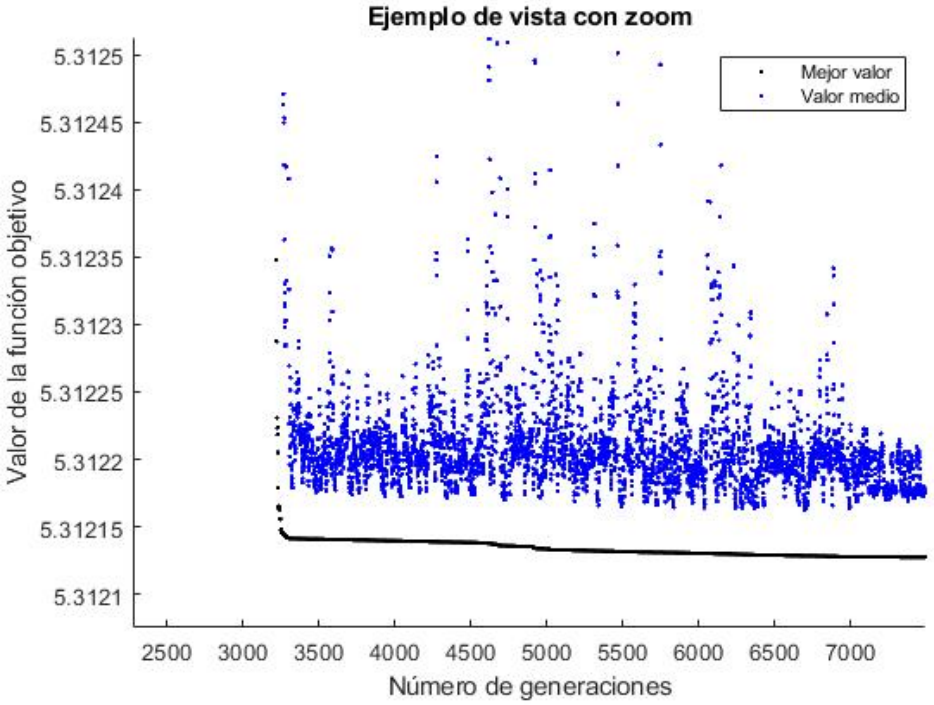


Figura A.2 Zoom en el eje *y* de la Figura A.1.

En el caso de no encontrar un individuo mejor, toda la población se irá pareciendo cada vez más y la diferencia entre la media de todos los individuos de la generación y el mejor de ellos se irá reduciendo. Puede verse este efecto en las Figuras A.2, donde entorno a la generación 4500 la media (puntos azules) es mayor que entorno a la 7500, debido a la tendencia del algoritmo hacia un mínimo local en este caso.

Apéndice B
Códigos de MATLAB utilizados

B.1 Cassini1

Para el correcto funcionamiento de los algoritmos es necesario descargar los documentos *mga.m*, *cassini1.m* y *cassini1.mat* de la página de la ESA [4] donde se publicaron los problemas. Una vez introducidos estos con los códigos que se presentan, podrán ejecutarse las optimizaciones sin ningún tipo de problema.

B.1.1 Algoritmo genético

Código B.1 Código de optimización del problema *Cassini1* con algoritmo genético.

```
% Carga del problema
load cassini1.mat
global MGAproblem

% Restricciones
x1=MGAproblem.bounds.lower';
x2=MGAproblem.bounds.upper';

tic

options_ga=optimoptions('ga');
options_ga1= optimoptions(options_ga,'MaxGenerations',7500,...
    'PlotFcn',{@gaplotbestf},...
    'MaxStallGenerations',7500,...
    'FunctionTolerance',0,...
    'MutationFcn',{@mutationadaptfeasible,0.02},...
    'CrossoverFraction',0.75,...
    'PopulationSize',1000,...
    'EliteCount',0.01*1000);

[X,FVAL,EXITFLAG,OUTPUT,POPULATION,SCORES]=ga(@(t)cassini1(t,MGAproblem)
    ,6,[],[],[],[],x1,x2,[],options_ga1);

tiempo=toc/60;
```

B.1.2 Enjambre de partículas

Código B.2 Código de optimización del problema *Cassini1* con algoritmo de enjambre de partículas.

```
% Carga del problema
load cassini1.mat
global MGAproblem

% Restricciones
x1=MGAproblem.bounds.lower';
x2=MGAproblem.bounds.upper';

tic

options_swarm=optimoptions('particleswarm');
options_swarm= optimoptions(options_swarm,'SwarmSize',1000,...
    'MaxStallIterations',4000,...
    'MaxIterations',4000,...
    'PlotFcn',@pswplotbestf,...
    'InertiaRange',[1 1],...
    'SelfAdjustmentWeight',0.75,...
    'SocialAdjustmentWeight',0.25);


[X,FVAL]=particleswarm(@(t)cassini1(t,MGAproblem),6,x1,x2,options_swarm)
    ;

tiempo=toc/60;
```

B.1.3 Algoritmo mixto

Código B.3 Código de optimización del problema *Cassini1* con algoritmo mixto.

```
% Carga del problema
load cassini1.mat
global MGAproblem

% Restricciones
x1=MGAproblem.bounds.lower';
x2=MGAproblem.bounds.upper';

tic, puntosbuenos=[]; n=0; X=[];

while n<10

%%%%% ALGORITMO GENETICO %%%%%
    options_ga=optimoptions('ga');
    options_ga1= optimoptions(options_ga,'MaxGenerations',300,...
```

```
    'PlotFcn',{@gaplotbestf},...
    'MaxStallGenerations',300,...
    'FunctionTolerance',0,...
    'MutationFcn',{@mutationadaptfeasible,0.02},...
    'CrossoverFraction',0.75,...
    'PopulationSize',1000,...
    'EliteCount',0.01*1000,...
    'InitialPopulationMatrix',X);

[X,FVAL,EXITFLAG,OUTPUT,POPULATION,SCORES]=ga(@(t)cassini1(t,MGAproblem)
    ,6,[],[],[],[],x1,x2,[],options_ga1);

% Se guardan los mejores valores de cada generación
fig_GA=gcf;
ejes_GA=fig_GA.Children(end);
puntosbuenos=[puntosbuenos ejes_GA.Children(2).YData];
close all


%%%%% OPTIMIZACIÓN POR ENJAMBRE DE PARTÍCULAS %%%%%
    options_swarm=optimoptions('particleswarm');
    options_swarm= optimoptions(options_swarm,'SwarmSize',1000,...
    'MaxStallIterations',200,...
    'InitialSwarmMatrix',X,...
    'MaxIterations',200,...
    'PlotFcn',@pswplotbestf,...
    'InertiaRange',[1 1],...
    'SelfAdjustmentWeight',0.75,...
    'SocialAdjustmentWeight',0.25);

[X,FVAL]=particleswarm(@(t)cassini1(t,MGAproblem),6,x1,x2,options_swarm)
    ;

% Se guardan los mejores valores de cada generación
fig_PSO=gcf;
ejes_PSO=fig_PSO.Children(4);
puntosbuenos=[puntosbuenos ejes_PSO.Children.YData];
close all

n=n+1;
end % Fin del bucle while

t=toc/60;

% Se representan los puntos correspondientes a los mejores valores de
    cada generación
plot(puntosbuenos,'.b')
ylabel('Valor de la función objetivo')
xlabel('Número de generaciones')
legend('Mejor valor')
```

B.2 GTOC1

Como en el problema anterior es necesario descargar *mga.m*, *gtoc1.m* y *gtoc1.mat* de la página de la ESA [4] para evitar errores durante la ejecución de los códigos.

B.2.1 Algoritmo genético

Código B.4 Código de optimización del problema *GTOC1* con algoritmo genético.

```
% Carga del problema
load gtoc1.mat
global MGAproblem

% Restricciones
x1=MGAproblem.bounds.lower';
x2=MGAproblem.bounds.upper';

tic

options_ga=optimoptions('ga');
options_ga1= optimoptions(options_ga,'MaxGenerations',7500,...
    'PlotFcn',{@gaplotbestf},...
    'MaxStallGenerations',7500,...
    'FunctionTolerance',0,...
    'MutationFcn',{@mutationadaptfeasible,0.02},...
    'CrossoverFraction',0.75,...
    'PopulationSize',1000,...
    'EliteCount',0.01*1000);

[X,FVAL,EXITFLAG,OUTPUT,POPULATION,SCORES]=ga(@(t)-gtoc1(t,MGAproblem)
    ,8,[],[],[],[],x1,x2,[],options_ga1);

 tiempo=toc/60;
```

B.2.2 Enjambre de partículas

Código B.5 Código de optimización del problema *GTOC1* con algoritmo de enjambre de partículas.

```
% Carga del problema
load gtoc1.mat
global MGAproblem

% Restricciones
x1=MGAproblem.bounds.lower';
x2=MGAproblem.bounds.upper';

tic
```

```
options_swarm=optimoptions('particleswarm');
options_swarm= optimoptions(options_swarm,'SwarmSize',1000,...
    'MaxStallIterations',4000,...
    'MaxIterations',400,...
    'PlotFcn',@pswplotbestf,...
    'InertiaRange',[1 1],...
    'SelfAdjustmentWeight',0.75,...
    'SocialAdjustmentWeight',0.25);

[X,FVAL]=particleswarm(@(t)-gtoc1(t,MGAproblem),8,x1,x2,options_swarm);

 tiempo=toc/60;
```

B.2.3 Algoritmo mixto

Código B.6 Código de optimización del problema *GTOC1* con algoritmo mixto.

```
% Carga del problema
load gtoc1.mat
global MGAproblem

% Restricciones
x1=MGAproblem.bounds.lower';
x2=MGAproblem.bounds.upper';

tic, puntosbuenos=[]; n=0; X=[];

while n<10

%%%%% ALGORITMO GENETICO %%%%%
    options_ga=optimoptions('ga');
    options_ga1= optimoptions(options_ga,'MaxGenerations',300,...
    'PlotFcn',{@gaplotbestf},...
    'MaxStallGenerations',300,...
    'FunctionTolerance',0,...
    'MutationFcn',{@mutationadaptfeasible,0.02},...
    'CrossoverFraction',0.75,...
    'PopulationSize',1000,...
    'EliteCount',0.01*1000,...
    'InitialPopulationMatrix',X);

[X,FVAL,EXITFLAG,OUTPUT,POPULATION,SCORES]=ga(@(t)-gtoc1(t,MGAproblem)
    ,8,[],[],[],[],x1,x2,[],options_ga1);

% Se guardan los mejores valores de cada generación
fig_GA=gcf;
ejes_GA=fig_GA.Children(end);
puntosbuenos=[puntosbuenos ejes_GA.Children(2).YData];
close all
```

```
%%%%% OPTIMIZACIÓN POR EJAMBRE DE PARTÍCULAS %%%%%
   options_swarm=optimoptions('particleswarm');
   options_swarm= optimoptions(options_swarm,'SwarmSize',1000,...
   'MaxStallIterations',4000,...
   'InitialSwarmMatrix',X,...
   'MaxIterations',200,...
   'PlotFcn',@pswplotbestf,...
   'InertiaRange',[1 1],...
   'SelfAdjustmentWeight',0.75,...
   'SocialAdjustmentWeight',0.25);

[X,FVAL]=particleswarm(@(t)-gtoc1(t,MGAproblem),8,x1,x2,options_swarm);

% Se guardan los mejores valores de cada generación
fig_PSO=gcf;
ejes_PSO=fig_PSO.Children(4);
puntosbuenos=[puntosbuenos ejes_PSO.Children.YData];
close all

n=n+1;
end % Fin del bucle while

t=toc/60;

plot(puntosbuenos,'.b')
ylabel('Valor de la función objetivo')
xlabel('Número de generaciones')
legend('Mejor valor')
```

B.3 Messenger

En este problema es necesario descargar los documentos *mga_dsm.m*, *messenger.m* y *messenger.mat* para poder ejecutar correctamente los códigos que siguen.

B.3.1 Algoritmo genético

Código B.7 Código de optimización del problema *Messenger* con algoritmo genético.

```
% Carga del problema
load messenger.mat
global MGADSMproblem

% Restricciones
x1=MGADSMproblem.bounds.lower';
x2=MGADSMproblem.bounds.upper';
```

```
tic

options_ga=optimoptions('ga');
options_ga= optimoptions(options_ga,'MaxGenerations',35000,...
    'PopulationSize',10000,...
    'PlotFcn',{@gaplotbestf},...
    'FunctionTolerance',0,...
    'MaxStallGenerations',35000,...
    'MutationFcn',{@mutationadaptfeasible, 0.02},...
    'CrossoverFraction',0.75,...
    'EliteCount',0.01*1000);

[X,FVAL,EXITFLAG,OUTPUT,POPULATION,SCORES]=ga(@(t)messenger(t,
    MGADSMproblem),18,[],[],[],[],x1,x2,[],options_ga);

tiempo=toc/3600;
```

B.3.2 Enjambre de partículas

Código B.8 Código de optimización del problema *Messenger* con algoritmo de enjambre de partículas.

```
% Carga del problema
load messenger.mat
global MGADSMproblem

% Restricciones
x1=MGADSMproblem.bounds.lower';
x2=MGADSMproblem.bounds.upper';

tic

options_swarm=optimoptions('particleswarm');
options_swarm= optimoptions(options_swarm,'SwarmSize',10000,...
    'MaxStallIterations',12500,...
    'MaxIterations',12500,...
    'PlotFcn',@pswplotbestf,...
    'InertiaRange',[1 1],...
    'SelfAdjustmentWeight',0.75,...
    'SocialAdjustmentWeight',0.25);

[X,FVAL]=particleswarm(@(t)messenger(t,MGADSMproblem),18,x1,x2,
    options_swarm);

tiempo=toc/3600;
```

B.3.3 Algoritmo mixto

Código B.9 Código de optimización del problema *Messenger* con algoritmo mixto.

```
% Carga del problema
load messenger.mat
global MGADSMproblem

% Restricciones
x1=MGADSMproblem.bounds.lower';
x2=MGADSMproblem.bounds.upper';


tic, n=0; X=[]; puntosbuenos=[];
while n<26

%%%% ALGORITMO GENETICO %%%%

    options_ga=optimoptions('ga');
    options_ga= optimoptions(options_ga,'MaxGenerations',300,...
    'PlotFcn',{@gaplotbestf},...
    'MaxStallGenerations',300,...
    'FunctionTolerance',0,...
    'MutationFcn',{@mutationadaptfeasible,0.02},...
    'CrossoverFraction',0.75,...
    'PopulationSize',10000,...
    'EliteCount',0.01*10000,...
    'InitialPopulationMatrix',X);

[X,FVAL,EXITFLAG,OUTPUT,POPULATION,SCORES]=ga(@(t)messenger(t,
    MGADSMproblem),18,[],[],[],[],x1,x2,[],options_ga);

% Se guardan los mejores valores de cada generación
fig_GA=gcf;
ejes_GA=fig_GA.Children(end);
puntosbuenos=[puntosbuenos ejes_GA.Children(2).YData];
close all


%%%%% OPTIMIZACIÓN POR ENJAMBRE DE PARTÍCULAS %%%%

    options_swarm=optimoptions('particleswarm');
    options_swarm= optimoptions(options_swarm,'SwarmSize',10000,...
    'MaxStallIterations',200,...
    'MaxIterations',200,...
    'PlotFcn',@pswplotbestf,...
    'InertiaRange',[1 1],...
    'SelfAdjustmentWeight',0.75,...
    'SocialAdjustmentWeight',0.25,...
```

```
    'InitialSwarmMatrix',X);

[X,FVAL]=particleswarm(@(t)messenger(t,MGADSMproblem),18,x1,x2,
    options_swarm);

% Se guardan los mejores valores de cada generación
fig_PSO=gcf;
ejes_PSO=fig_PSO.Children(4);
puntosbuenos=[puntosbuenos ejes_PSO.Children.YData];
close all

n=n+1

end %Fin del bucle while

tiempo=toc/3600;

% Se representan los puntos correspondientes a los mejores valores de
    cada generación
plot(puntosbuenos,'.b')
ylabel('Valor de la función objetivo')
xlabel('Número de generaciones')
legend('Mejor valor')
```

B.4 Algoritmo aleatorio

Código B.10 Código del algoritmo utilizado para confrontar los resultados. Aplicado a *Cassini1*.

```
% Carga del problema
load cassini1.mat
global MGAproblem

% Restricciones
x1=MGAproblem.bounds.lower';
x2=MGAproblem.bounds.upper';

Generaciones=7500;
Poblacion=1000;
Num_eva=Generaciones*Poblacion;
intervalo=x2-x1;
J=zeros(Generaciones, Poblacion);

for k=1:Num_eva
    x=x1+rand(length(x1),1).*intervalo;
    J(k)= cassini1(x,MGAproblem);
end
Mejor=min(min(J));
```

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] J. C. Bansal, P. K. Singh, Mukesh Saraswat, Abhishek Verma, Shimpi Singh Jadon, and Ajith Abraham, *Inertia weight strategies in particle swarm optimization*, 2011 Third World Congress on Nature and Biologically Inspired Computing, 2011, pp. 633–640.

[2] Fernando Sancho Caparrini, *Pso: Optimización por enjambre de partículas*, http://www.cs.us.es/~fsancho/?e=70, Online; acceso 12 de julio 2021.

[3] R.C. Eberhart and Yuhui Shi, *Tracking and optimizing dynamic systems with particle swarms*, Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546), vol. 1, 2001, pp. 94–100 vol. 1.

[4] ESA, *Global trajectory optimisation problems*, https://www.esa.int/gsp/ACT/projects/gtop/, Online; acceso 12 de julio 2021.

[5] R. Hinterding, *Gaussian mutation and self-adaption for numeric genetic algorithms*, Proceedings of 1995 IEEE International Conference on Evolutionary Computation, vol. 1, 1995, pp. 384–.

[6] Khalid Jebari, *Selection methods for genetic algorithms*, International Journal of Emerging Sciences **3** (2013), 333–344.

[7] A Nikabadi and M Ebadzadeh, *Particle swarm optimization algorithms with adaptive inertia weight: A survey of the state of the art and a novel method*, IEEE journal of evolutionary computation (2008).

[8] Juan Murube Pemán, *El modelo heliocéntrico actual*, Diciembre 2016.

[9] Adam Piotrowski, Jarosław Napiórkowski, and Agnieszka Piotrowska, *Population size in particle swarm optimization*, Swarm and Evolutionary Computation **58** (2020), 100718.

[10] Riccardo Poli, James Kennedy, and Tim Blackwell, *Particle swarm optimization: An overview*, Swarm Intelligence **1** (2007).

[11] Colin Reeves, *Genetic algorithms*, vol. 146, pp. 109–139, 09 2010.

[12] Y. Shi and R. Eberhart, *A modified particle swarm optimizer*, Evolutionary Computation Proceedings 1998. IEEE World Congress on Computational Intelligence, 2002, pp. 69–73.

[13] José Utreras, *Cassini - huygens mission trajectory to saturn*, https://www.behance.net/gallery/89042123/Cassini-Huygens., Online; acceso 12 de julio 2021.

[14] Rafael Váquez Valenzuela, *Astronáutica y vehículos espaciales*, Apuntes de clase (ETSI Sevilla), extraídos de http://aero.us.es/astro/desc.html (2020-2021).

[15] T. Vinkó and D. Izzo, *Global optimisation heuristics and test problems for preliminary spacecraft trajectory design*, September 2008.

[16] Jianbin Xin, Guimin Chen, and Yubao Hai, *A particle swarm optimizer with multi-stage linearly-decreasing inertia weight*, 2009 International Joint Conference on Computational Sciences and Optimization, vol. 1, 2009, pp. 505–508.